TEMEL SEVİYE PYTHON

❓ *Bu döküman, Python öğrenmek isteyen kişilerin öğrenimine katkı sağlamak amacı ile hazırlanmıştır.*

⚠ *Bu (1) döküman, tamamen Python öğrenmenizi garantileyemez.*

✅ *Bu döküman baz alınarak,hazırlayacağınız eğitimlerde, bu kitaptan kesitlere yer verebilirsiniz*

Python kodlarınızı yazmak için, belirli bir noktaya gelene kadar online editörleri kullanabilirsiniz . Bunun için, googleye "Python Online Compiler" yazmanız yeterli olacaktır . Daha sonrasında PyCharm veya VSCode kullanabilirsiniz

Telegram: @ucretsizegitim

YouTube: Teknoloji TVv

```
x=5
y="hello world"
print(x)
```

Kodlar

```
5
```

Çıktı

```
sayi1=15
if(sayi1<2):
    print("sayi1 küçüktür 2")
else:
    print("sayi1 büyüktür 2")

#print kullanmadan çıktı verir
```

```
      sayi1 büyüktür(>) 2
```

şeklindedir

Bilgilendirmeler

(Diyez işareti) : Yorum satırı oluşturmak için kullanılır.Yorum satırlarını program çalıştırılmaz

```
x="yusuf"
x=str(3)

# x değişkeninin yeni karşılığı 3 olacak

print(x) #print,sonucu yazdırmak için kullanılır
```

```
3
```

```
y=15
y=int(12)
#y'nin ilk değeri olan 15'i , 12 ile değiştirdik

print(y) #ekranda 12 değerini görmeliyiz
```

```
z=35,5
#z'yi 35,5 olarak ayarladık

z=float(36)
#z'nin 35,5 olan değerini 36 ile değiştirdik

print(z)
#z değişkenini yazdırdık ve 36 çıktısını aldık
```

```
x="yusuf"
y=15
print(type(x))
#x değişkeninin karşılığındaki atamanın, #değişken
türünü alır

# print(type(y)) #x değişkeninin karşılığındaki
#atamanın, değişken türünü alır
#yazdırınca çıktısı aşağıdaki gibi olacak
<class 'str'>
<class 'int'>
```

```
isim='yusuf'

#  çift tırnak ile tek tırnak arasında fark yoktur,
yine yusuf diye çıktı verir

print(isim)
```

```
meyve1,meyve2,meyve3="elma","armut",
"portakal"

print (meyve1+" "+meyve2+" "+meyve3)

elma armut portakal
```

```
meyve1 = meyve2 = meyve3 = "hıyar"
print(meyve1)

hıyar
```

```
sebzeler=["karpuz","kavun","limon"]
karpuzsebzesi,kavunsebzesi,limonsebzesi=sebzeler
print(karpuzsebzesi)
print(kavunsebzesi)
print(limonsebzesi)

#aşağıdaki şekilde çıktı alırız
```

```
karpuz
kavun
limon
```

```
BencePython="harika"
print( "python bir " + BencePython)
```

```
python bir harika
```

```
sayi1=11
sayi2=22
print(sayi1+sayi2)
```

```
33
```

```
def paket():
    print("fonksiyon çalıştığında, içerisindeki kodlar
çalışır")

paket()
```

```
#paket() : paket adındaki fonksiyonu çağırır ve
çalıştırır
    #def = fonksiyon , paket adında fonksiyon
tanımladık
    # fonksiyon: belirli bir kod bloğunu sadece 1
kere tanımlayıp ihtiyacımız olduğu yerlerde
çağırabilmemizi sağlar
```

global nedir : Global anahtar kelime, kullanıcının geçerli kapsamın dışındaki bir değişkeni değiştirmesine izin veren bir anahtar kelimedir.

Global olmayan bir kapsamdan, yani bir fonksiyonun içinden global değişkenler oluşturmak için kullanılır.

Global anahtar kelime bir fonksiyoniçerisinde sadece atama yapmak istediğimizde veya bir değişkeni değiştirmek istediğimizde kullanılır

Text(Metin) Type: str

Numeric(Sayısal) Types: int, float, complex

Sequence(Sıra) Types: list, tuple, range

Mapping(Haritalama) Type: dict

Set(Ayarlama,Dizi) Types: set, frozenset

Boolean(Doğru-Yanlış) Type: bool

Binary(İkili) Types: bytes, bytearray, memoryview

Range: aralık, range() işlevi, verilen başlangıç tamsayısından bitiş tamsayısına kadar, değişmez sayı dizisini oluşturur.

Bir for döngüsü kullanarak yineleyebileceğimiz bir dizi tamsayıdan oluşan bir aralık nesnesi döndüren yerleşik bir işlevdir.

```
deger1=1
deger2=2.9
deger3=3j



print(type(deger1)) # int=tam sayı
print(type(deger2)) # float=virgüllü
print(type(deger3)) # complex=karmaşık
```

```
sayi1=1j
sayi2= 3+5j
sayi3=-5j

print(type(sayi1))
#sayi1'in karşısındaki değerin tipi:complex değer

print(type(sayi2)) #complex değer

print(type(sayi3)) #complex değer
```

```
import random
#random sınıfını entegre ediyoruz

print(random.randrange(1,10))

# rand aralığını belirtiyoruz 1-10 arası rastgele
# bir sayı verecek
```

```
cokluSatir = """üç tırnak
arasına
yazılır """

print(cokluSatir)
```

```
degiskenimiz="Yusuf "
print(degiskenimiz[1])
```

```
# degiskenimizin 1 no'lu karakterini getirir
# en soldaki 0'dır ve sağa doğru artarak ilerler

# [1] = u harfini getirir
# [0] = y harfini getirir
```

```
for degisken in "elma":
    print(degisken)
```

```
e
l
m
a
```

```
# for : elma kelimesini döngü haline getirdi
```

```
degisken="Yusuf"
print(len(degisken))
```

```
# len = uzunluk
# degisken'e atanan kelimenin harf sayısını getirir
```

```
metin="Bu metinden, hangi kelimeyi
arayacağımızı belirteceğim"
print("hangi" in metin)
```

```
# hangi kelimesini, metin değişkeni içinde ara,
# var ise True ,
# yok ise False değeri döndür/ekrana yaz
```

```
metnimiz="Bu metinden hangi kelimeyi
arayacağımızı belirteceğim"

if("hangi" in metnimiz):
    print("Evet! 'hangi' kelimesi, metnimiz içerisinde
bulunuyor.")

else:
    print("Hayır! 'hangi' kelimesi, metnimiz
içerisinde bulunmuyor")
```

```
metnimiz="Bu metinden hangi kelimeyi arayacağımızı belirteceğim"

print("hangi" not in metnimiz)

#hangi kelimesini, metnimiz değişkeni içerisinde bulamadın mı?
```

hayır

```
#fonksiyonun olumsuz olmasına dikkat ediniz
```

```
metin="Bu metinden hangi kelimeyi arayacağımızı belirteceğim"

if "araba" not in metin:
    print("Evet,metin değişkeninin içinde 'Araba' kelimesi yok")
```

```
degisken="Redkit"
print(degisken[2:5])
```

dki

```
# indexler : R:0 , E:1 , D:2 , K:3 , İ:4, T:5
# slicing:dilimleme/kesme
```

```
degiskenimiz="Redkit"
print(degiskenimiz[:5])
```

```
Redki
```

```
# :5 'ya dikkat ediniz
# : kesmek için kullanılır
```

```
degisken="Redkit"
print(degisken[2:])
```

```
#soldan 2 kesti
#sağda değer verilmediği için değişiklik olmadı
```

```
dkit
```

```
degiskenimiz="Merhaba Dünya"
print( degiskenimiz  [-5 : -2] )
```

```
Dün
```

```
degiskenim="Bursa"
print(degiskenim.upper())
```

```
BURSA
```

```
#upper:büyük harfe çevirir
```

```
degiskenimiz="ANTalya"
print(degiskenimiz.lower())
```

```
antalya
```

```
# strip = soymak,çıkarmak

metnimiz=" Bu metnin başında ve sonunda bir
boşluk bırakıldı "
print(metnimiz.strip())
```

```
Bu metnin başında ve sonunda bir boşluk bırakıldı
```

```
#strip fonksiyonu boşlukları kaldırdı
```

```
#replace=değişim,dönüşüm

degisken="Merhaba Dünya"
print(degisken.replace("h","a"))
```

```
Meraaba Dünya
```

```
degiskenimiz="Python Öğreniyorum"
print(degiskenimiz.split(","))

# degiskenimiz içerisindeki veriden
# iki tırnak arasındaki değeri çektik

'Python Öğreniyorum'
```

```
degisken1="Ali"
degisken2="Rıza"
birlestir=degisken1 + " " + degisken2
print(birlestir)
Ali Rıza
```

```
yasiniz=36
metin="Benim adım Burhan, ben {} yaşındayım"
print(metin.format(yasiniz))

Benim adım Burhan, ben 36 yaşındayım
```

```
#quantity:miktar
#item:öğe
#price:fiyat
#myOrder:siparişim
miktar=3
ogeNumarasi=567
fiyat=49.95
siparisim=\
    "Ben: {} tane, {} numarali öğeyi, {}₺'ye istiyorum"

print(siparisim.format(miktar,ogeNumarasi,fiyat))
```

```
Ben: 3 tane, 567 numarali öğeyi, 49.95₺'ye istiyorum
```

```
ciftTirnakKullanimi="Tırnak içerisinde tekrar tırnak
kullanmak mümkün, \" 2.tırnak içi \" şeklinde
kullanmalısınız "

print(ciftTirnakKullanimi)
```

```
Tırnak içerisinde tekrar tırnak kullanmak mümkün, "
2.tırnak içi " şeklinde kullanmalısınız
```

```
metin = "Bu örnekte capitalize fonksiyonunu
göreceğiz"

fonksiyon = metin.capitalize()
print (fonksiyon)

#capitalize:İlk karakteri büyük harfe dönüştürür
```

```
metin = "Tüm Karakterleri Küçük Harfe Çevirir"
fonksiyon = metin.casefold()
print(fonksiyon)

tüm karakterleri küçük harfe çevirir

#casefold: küçük harfe çevirir
```

```
fonksiyon = metnimiz.center(20)
#20px boşluk alıyor
print(x)

        Muz

#center:ortalamayı sağlar
```

```
metin = "Ben Elma severim , elma güzel bir
meyvedir"
fonksiyon = metin.count("elma")
print(fonksiyon)
```

```
1
```

#Büyük-küçük harfe duyarlı olduğu için sadece aynı olanları baz alır

#count:miktar

```
metnimiz = "Benim adım name"
fonksiyon = metnimiz.encode()
print(fonksiyon)
```

```
b'Benim ad\xc4\xb1m name'
```

#encode:kodlamak

```
kontrolMetni="Cümlenin sonundaki noktayı kontrol edeceğiz ."
kontrol=kontrolMetni.endswith(".")
print(kontrol)
```

```
True
```

```
#endswith:Dize belirtilen değerle bitiyorsa true döndürür
#endswith() = True ya da False değer döndürür
```

```
#True=Doğru
#False=Yanlış
#Noktalama yerine son 2 kelime de örnek olarak kontrol ettirilebilir.True döndürecektir
```

```
tabBoslugu = "\tM\te\tr \t h\t aba"
bosluk =  tabBoslugu.expandtabs(1) #1 boşluk

print(bosluk)
```

```
M e r   h  aba
```

```
#expandtabs fonksiyonu, tab boşluğu bırakmayı sağlar
```

```
metin = "Find fonksiyonunu öğreneceğiz"
bul = metin.find("fonksiyonunu")
print(bul) #baş harfinin konumunu getirdi
```

```
5
```

```
yeniBul=metin.find("n")
print(yeniBul)
```

```
2
```

```
#saymaya soldan başlar
#en sol [0] , sağa doğru birer +artar
#F=0 , i=1 , n=2 ...
#find:bulmak
```

```
fiyatMetni = "Bu ürün sadece {fiyat:.3f} lira!"
print(fiyatMetni.format(fiyat = 1))
```

```
Bu ürün sadece 1.000 lira!
```

```
#.3f = .000

#format:biçim
```

```
metin1="Adım {isim} , ben {yas} yaşındayım"\
.format(isim="Yakup",yas=20)

print(metin1)
```

```
Adım Yakup , ben 20 yaşındayım
```

```
# \ alt satırdan devam etmeyi sağlar
```

```
metin2="Demirciye {item} artı basmaya {hedef}
gidiyorum"\
.format(item="dolunay kılıcımı",hedef="2.köye")

print(metin2)
```

```
Demirciye dolunay kılıcımı artı basmaya 2.köye
gidiyorum
```

```
#format:Bir dizede belirtilen değerleri biçimlendirir
```

```
metin="Metnin neresinde hoşgeldin kelimesinin
bulunduğunu gösterecek"

bul=metin.index("hoşgeldin")

print(bul)
```

```
17
```

```
metin="Metinde geçen e harfinin indexini bulacağız"

bul=metin.index("e")
#e harfini ilk bulduğun index sırası kaçta

print(bul)
```

```
1
```

```
bul=metin.index("e",5,10)
#5 ile 10 indexleri arasında e harfi kaçıncı sırada

print(bul)
```

```
6
```

```
#not:değeri bulamadığı zaman -1 değer döndürür
```

```
alfasayisalMetin = "Metin2"
kontrol = alfasayisalMetin.isalnum()
print(kontrol)
```

```
True
```

```
#alfasayısal:harfler, sayılar ve noktalama işaretleri
# gibi diğer karakterlerden oluşan

#metinde a-z harfler ve/veya 0-9 sayılar bulunup
#bulunmadığını kontrol eder,varsa True,yoksa False
döndürür

#alfasayısal olmayan karakterlere
#örnek olarak (boşluk) , !, # , % , & , ?
#gösterilebilir,aşağıdaki örnekteki gibi false
döndürürler
```

```
alfasayisalMetin = "Metin 2"
kontrol = alfasayisalMetin.isalnum()
print(kontrol)
```

```
False
```

```
#Alfasayısal olmayan karakter(boşluk) kullanıldığı
için, false değeri döndürdü
```

#örnek 1

```
metnimiz = "Pythonogreniyorum"
kontrol = metnimiz.isalpha()
print(kontrol)
```

```
True
```

#isalpha=hepsi alfabede bulunuyor mu

#örnek 2

```
metnimiz2 = "Python ogreniyorum"
kontrol2 = metnimiz2.isalpha()
print(kontrol2)
```

```
False
```

#boşluk alfabede bulunmaz dolayısıyla false döndürdü

```
metnimiz="123450a"
kontrol=metnimiz.isdigit()
print(kontrol)
```

```
False
```

#a değeri olmasaydı True dönecekti

#isdigit:hepsi sayı mı ,bunun kontrolünü yapar

# örnek 1

```
metin= "\u0033"
#bu 3 sayısının unicode şifrelemesi

kontrol=metin.isdecimal()
#bu da ondalıklı sayı kontrolü

# print(kontrol)
```

```
True
```

```
#ondalıklı sayı karakterleri 0-9 arasındaysa
#virgül veya noktaya dikkat edilmeden True döndürür
```

# örnek 2

```
metin2= "\u9258"
#rastgele yazdım

kontrol2=metin2.isdecimal()
print(kontrol2)
```

```
False
```

```
#Rastgele yazdığımız bu unicode'nin bir karşılığının
olmadığını gördük
```

```
deyim1="Deneme"
deyim2="Deneme0101"
deyim3="0101"
deyim4="abc"
deyim5="ab cd"
```

```
kontrol=deyim1.isidentifier()
kontrol1=deyim2.isidentifier()
kontrol2=deyim3.isidentifier()
kontrol3=deyim4.isidentifier()
kontrol4=deyim5.isidentifier()
```

```
print(kontrol)
print(kontrol1)
print(kontrol2)
print(kontrol3)
print(kontrol4)
```

```
True
True
False
True
False
```

```
#identify:tanımlamak
#identity:kimlik
```

```
metin = "Bu metinde bütün karakterler büyük değil"
kontrol = metin.islower()
print(kontrol)
```

```
False
```

```
#islower() fonksiyonu,metnin tüm karakterlerinin, küçük harfle yazılıp
#yazılmadığını kontrol edip , eğer küçükse True , değilse False döndürür
```

```
metnimiz="0555 559 59 59"
kontrol=metnimiz.isnumeric()
print(kontrol)
```

```
False
```

```
#Boşluk kullanıldığı için false döndürdü
#05555595959 şeklinde yazsaydık True döndürecekti
```

```
# örnek 1
metnimiz="Bu metinde yazdırılabilirliği kontrol edecek fonksiyonu öğreneceğiz"
kontrol=metnimiz.isprintable()
print(kontrol)
```

```
True
```

```
# örnek 2
metnimiz2="Bu metinde\nde yazdırılabilirliği kontrol edecek fonksiyonu öğreneceğiz"
kontrol2=metnimiz2.isprintable()
print(kontrol2)
```

```
False
```

```
#isspace:Dizedeki tüm karakterler boşluksa True
döndürür
```

#örnek 1

```
metin="isspace fonksiyonunu öğreneceğiz"
kontrol=metin.isspace()
print(kontrol)
```

```
False
```

#örnek 2

```
metin2="    "
kontrol2=metin2.isspace()
print(kontrol2)
```

```
True
```

```
metin="Her Kelimenin Büyük Harfle Başladığını
Kontrol Eden Fonksiyon İstitle()"

kontrol=metin.istitle()
print(kontrol)
```

```
True
```

```
metin="METİNDEKİ TÜM HARFLER BÜYÜK MÜ?"

kontrol=metin.isupper()
print(kontrol)
```

```
True
```

```
# .join fonksiyonu, belirlediğiniz karakteri, ayırıcı olarak kullanmayı sağlar"
```

```
isimler="Ali","Ahmet","Mehmet"

fonksiyon= "/".join(isimler)

# print(fonksiyon)
```

```
Ali/Ahmet/Mehmet
```

#örnek 1

```
kelime = "Araba"
fonksiyon = kelime.ljust(20)
#kelimeden sonra, 20 karakter boşluk bırakılmasını
istedik
print(fonksiyon, "hızla yoldan geçti")
```

```
Araba                hızla yoldan geçti
```

#örnek 2

```
kelime="Araba"
fonksiyon=kelime.ljust(20,"0")
#20 = totalde 20 karakter olacak,
#Araba=5 karakter, kalan 15 karakteri 0 yazacak
print(fonksiyon)
```

```
Araba000000000000000
```

```
metin="Bu örnekte lower fonksiyonunu
öğreneceğiz"
fonksiyon=metin.lower()
print(fonksiyon)
```

```
bu örnekte lower fonksiyonunu öğreneceğiz
```

```
#lower() : metin'e atanan bütün karakterleri küçük
harfe çevirir

#Semboller ve Sayılar yoksayılır
```

```
metin="   Bu örnekte lstrip fonksiyonunu öğreneceğiz   "
fonksiyon=metin.lstrip()
print(fonksiyon)
```

#L=left
#lstrip bu örnekte: metnimiz'in sol tarafındaki,baştaki boşluğu kaldırdı

```
metin=",,,,Bu örnekte de lstrip() fonksiyonunu görüyoruz..." # lstrip örnek 1
fonksiyon=metin.lstrip(",")
print(fonksiyon)
```

```
Bu örnekte de lstrip() fonksiyonunu görüyoruz...
```

#virgülleri kaldırdık
#virgül yerine boş () bıraksaydık, soldan boşlukları kaldıracaktı

```
metin2=",,,,Bu örnekte de lstrip() fonksiyonunu görüyoruz..."  # lstrip örnek 2
fonksiyon2=metin2.lstrip(".")
print(fonksiyon2)
```

```
,,,,Bu örnekte de lstrip() fonksiyonunu görüyoruz...
```

#lstrip sadece sol taraftaki karakterleri baz alır o yüzden noktalarda
#bir değişim olmadı

```
#translate=dönüştürme,çevirme
```

```
metin="Bu örnekte translate() fonksiyonunu öğreneceğiz"


fonksiyon=metin.maketrans("ö", "o")
print(metin.translate(fonksiyon))
```

```
Bu ornekte translate() fonksiyonunu oğreneceğiz
```

```
#fonksiyonumuz, metinimizin içeriğini dönüştürecek


# ö'yü o'ya dönüştürmesini istedik


#yazdır(metnimizin fonksiyondan çıktığı halini
yazdırdık)
```

```
metnimiz="Bu örnekte de translate fonksiyonunu
göreceğiz"

degisken1="nfo" #n=[0'ıncı index] #f=[1. index]


degisken2="nfö" #n=[0'ıncı index] #f=[1. index]


#aynı index değerine sahip olan[2.index]
karakterler, fonksiyondan farklı çıktı

fonksiyon=metnimiz.maketrans(degisken1,
  degisken2)

degisken1 = x
degisken2 = y

#y'den önce bir boşluk bıraktığımızı görmüşsünüzdür

print(metnimiz.translate(fonksiyon))
```

```
Bu örnekte de translate fönksiyönunu göreceğiz
```

```
metnimiz="Bu örnekte de translate fonksiyonunu göreceğiz"


degisken1="fonk" #metinde değişecek kısım
degisken2="fönk" #değiştikten sonra istenilen
degisken3="ö" #istenilenden sonra istenmeyen karakterler


fonksiyon=metnimiz.maketrans(degisken1,degisken2,degisken3)

print(metnimiz.translate(fonksiyon) )
```

```
Bu rnekte de translate fönksiyönunu greceğiz
```

```
# yazdır
# neyi:metnimizi
# metnimizi nasıl yazdıracak: .translate fonksiyonu kullanarak
# en son fonksiyondan geçir ve sonucu ekrana yazdır
# partition() örnek 1
metnimiz="Benim en sevdiğim renk sarı"
fonksiyon=metnimiz.partition("renk")
print(fonksiyon)
```

```
('Benim en sevdiğim ', 'renk', ' sarı')
```

```
# partition() örnek 2
metnimiz="Benim en sevdiğim renk sarı"
fonksiyon=metnimiz.partition("sevdiğim")
print(fonksiyon)
```

```
('Benim en ', 'sevdiğim', ' renk sarı')
```

```
#partition:üç öğe içeren bir demete böler
#1.bölünen, belirlenen kelimenin öncesi
#2.bölünen, belirlenen kelime
#3.bölünen, belirlenen kelimenin sonrası
```

```
metnimiz="Ben yazılımı seviyorum"
fonksiyon=metnimiz.replace("yazılımı","satrancı")
print(fonksiyon)
```

```
Ben satrancı seviyorum
```

```
#replace():belirlenen kelimeleri,istenilen kelimeye
dönüştürür
```

```
#3 kere Ben yazılımı seviyorum yazdık
metnimiz="Ben yazılımı seviyorum,Ben yazılımı seviyorum,Ben yazılımı seviyorum"

#yazılımı kelimesini,satrancı kelimesine çevirdik,sadece ilk 2
#yazılımı kelimesini,satrancı ile değiştirmesini istedik

fonksiyon=metnimiz.replace("yazılımı","satrancı",2)

print(fonksiyon)
```

```
Ben satrancı seviyorum,Ben satrancı seviyorum,Ben yazılımı seviyorum
```

```
metnimiz="Bu örnekte rfind fonksiyonunu göreceğiz"
fonksiyon=metnimiz.rfind("Bu")
print(fonksiyon)
```

```
0
```

```
#Bu = [0] , Örnekte = [1] , rfind= [2] ...

# rfind() : Dizeyi belirtilen bir değer için arar ve bulunduğu yerin son konumunu döndürür
```

```
metnimiz="Bu örnekte de rfind fonksiyonunu
göreceğiz"
fonksiyonumuz=metnimiz.rfind("e")
print(fonksiyonumuz)

#çıktısı:38

#e'nin son konumu, 38'inci indexte yer alıyormuş
#boşluklar dikkate alınmaz
```

```
metnimiz="Bu örnekte de rfind fonksiyonunu
göreceğiz"

fonksiyonum=metnimiz.rfind("e",5,10)

#5'inci ve 10'uncu indexler arasında kaçıncı indexte
"e" bulunuyor?

print(fonksiyonum)

9

#B=[0.index]
#u=[1.index]
#ö=[2.index]...
```

```
metnimiz="Bu örnekte de rfind fonksiyonunu
göreceğiz"

#fonksiyonu,print içine gömebiliriz

print(metnimiz.rfind("q")) #çıktı:-1
print(metnimiz.rindex("q")) #hata oluşturdu
```

```
metin="Bu örnekte rindex() fonksiyonunu
göreceğiz"
fonksiyon=metin.rindex("örnekte")
print(fonksiyon)
```

```
3
```

```
#göreceğiz:[0]
#fonksiyonunu:[1]
#rindex():[2]
#örnekte:[3]



#rindex=index numarasını (sağdan) saymaya başlar
```

```
metnimiz="Bu örnekte" #10 index
fonksiyon=metnimiz.rjust(20)

# 20-10=10 boşluk (sola) bırak , sağına yaz

print(fonksiyon," rjust() fonksiyonunu ele
alacağız")
```

```
          Bu örnekte  rjust() fonksiyonunu ele alacağız
```

```
#10 boşluk bırak ve yazmaya başla(metnimiz)
# rjust(x kadar boşluk) ve metni sağa yaslayarak
yazdır
```

```
metnimiz="Bu örnekte"
#10 index

fonksiyon=metnimiz.rjust(15,"x")
print(fonksiyon)
```

```
xxxxxBu örnekte
```

```
#Bu örnekte : 10 index
#İstenilen 15 index
#x : 5 index
```

```
metin="Bu örnekte splitlines\nfonksiyonunu göreceğiz"
fonksiyon=metin.splitlines()
print(fonksiyon)
```

```
['Bu örnekte splitlines', 'fonksiyonunu göreceğiz']
```

```
#split:bölünmüş
#lines:çizgiler,şeritler
#varsayılan olarak splitlines(False) alır
#yani bölünmeler(\n) görünmesin demektir
#Görünmesini istersek (True) aşağıdaki gibi olur
```

```
metin="Bu örnekte splitlines\nfonksiyonunu göreceğiz"
fonksiyon=metin.splitlines(True)
print(fonksiyon)
```

```
['Bu örnekte splitlines\n', 'fonksiyonunu göreceğiz']
```

```
#splitlines:Dizeyi satır sonlarında böler ve bir liste döndürür
```

```
metin="Bu örnekte startswith fonksiyonunu
inceleyeceğiz"
fonksiyon=metin.startswith("Bu")
print(fonksiyon)
```

```
True
```

```
#startswith: değişkenin , fonksiyon içerisindeki değer
ile başlayıp başlamadığını kontrol eder
```

```
metnimiz="Bu örnekte startswith fonksiyonunu
inceleyeceğiz"
fonksiyon=metnimiz.startswith("start",11,20)
print(fonksiyon)
```

```
True
```

```
#11. index : s
#20.index : h
#11.index ile 20.index aralığı, start ile mi başlıyor onu
kontrol ettirdik
```

```
metin = "     stripp     "

fonksiyon = metin.strip()
print("Bu örnekte", fonksiyon, "fonksiyonunu
inceleyeceğiz")
```

```
Bu örnekte stripp fonksiyonunu inceleyeceğiz
```

```
metin = "Bu ÖRNEKTE swapcase Fonksiyonunu
inceleyeceğiZ"

fonksiyon = metin.swapcase()

print(fonksiyon)
```

```
bU örnekte SWAPCASE fONKSIYONUNU
INCELEYECEĞIz
```

```
#swapcase : büyük harf girdilerini küçük harfe, küçük
harfleri de büyük harfe çevirir
```

```
degisken = "50"
fonksiyon = degisken.zfill(10)
print(fonksiyon)
```

```
0000000050
```

```
#8 tane sıfır
```

```
meyveler = ["elma", "armut", "çilek"]
meyveler.append("portakal")
print(meyveler)
```

```
['elma', 'armut', 'çilek', 'portakal']
```

```
#append fonksiyonu,listenin sonuna öğe ekler
```

```
liste1 = ["1.1", "1.2", "1.3"]
liste2 = ["2.1", "2.2", "2.3"]
liste1.append(liste2)
print(liste1)
['1.1', '1.2', '1.3', ['2.1', '2.2', '2.3']]
#appened() : listenin sonuna ekler
```

```
liste = ["1.2", "1.2", "1.3"]
liste.clear()
print(liste)
 [ ]
#clear() : temizler,siler
```

```
degisken = ["degisken1", "degisken2",
"degisken3"]

kopyala = degisken.copy()

print(kopyala)

['degisken1', 'degisken2', 'degisken3']

#copy() : listenin bir kopyasını döndürdü
```

```
loremipsum = ["dizgi", "sahte", "sahte" , "mıgır"]
say = loremipsum.count("sahte")
print(say)
```

```
2
```

```
#count:miktar
```

```
sayilar = [1, 9, 2, 9, 7, 8, 9, 3, 1]
fonksiyon = sayilar.count(9)
print(fonksiyon)
```

```
3
```

```
meyveler = ['vişne', 'ahududu', 'nar']
meyveler.insert(1, "kayısı")
print(meyveler)
```

```
['vişne', 'kayısı', 'ahududu', 'nar']
```

```
#meyveler değişkeninde , ilk durumda ahududu = [1]
#meyveler değişkeni , fonksiyondan çıktıktan sonra
kayısı = [1] oldu
```

```
ogeler = ['index0', 'index1', 'index2']
ogeler.pop(2)       #index 2'yi sil
print(ogeler)
```

```
['index0', 'index1']
```

```
ogeler = ['öğem', 'öğen', 'öğemiz']
ogeler.remove("öğemiz")
print(ogeler)

['öğem', 'öğen']
```

```
ogeler = ['öğe1', 'öğe2', 'öğe3' , 'öğe4' ]
ogeler.reverse()
print(ogeler)
['öğe4', 'öğe3', 'öğe2', 'öğe1']
# reverse : listeyi ters çevirir
```

```
alfabetiksirala = ['Türk', 'Öğün', 'Çalış' , 'Güven' ]
alfabetiksirala.sort()
print(alfabetiksirala)

['Güven', 'Türk', 'Çalış', 'Öğün']

#sort() : liste öğelerini alfabetik sıralamaya koyar
```

```
degisken = ['1', '2', '3']
degisken.sort(reverse=True)
print(degisken)

['3', '2', '1']

#degiskenleri alfabetik sıraladı, daha sonra ters
çevirdi
```

```
def Fonksiyonumuz(e):
  return len(e)

listemiz = ['1234', '12', '123', '1']
listemiz.sort(key=Fonksiyonumuz)
print(listemiz)
```

```
['1', '12', '123', '1234']
```

```
#listenin elemanlarını, harf sayısına göre sıraladık
#def:dinamik fonksiyon oluşturmayı sağlar
```

```
def Fonksiyonum(e):
  return e ['yıl']

Arabalar = [
  {'1)': 'Devrim', 'yıl': 2005},
  {'2)': 'Anadol A1' , 'yıl': 2000},
  {'3)': 'Tofaş Murat 124', 'yıl': 2013},
  {'4)': 'Oyak-Renault 12', 'yıl': 2011}
]
Arabalar.sort(key=Fonksiyonum)
print(Arabalar)
```

```
[{'2)': 'Anadol A1', 'yıl': 2000},
{'1)': 'Devrim', 'yıl': 2005},
{'4)': 'Oyak-Renault 12', 'yıl': 2011},
{'3)': 'Tofaş Murat 124', 'yıl': 2013}]
```

```
def Fonksiyonum(e):
  return len(e) #len:uzunluk
Arabalar = ['Anadol', 'Tofaş', 'Oyak' , 'Devrim']
Arabalar.sort(reverse=True, key=Fonksiyonum)
print(Arabalar)
```

```
['Anadol', 'Devrim', 'Tofaş', 'Oyak']
```

#len : uzunluk

#len : aksi belirtilmedikçe kısadan uzuna doğru sıralar

#Bu örnekte reverse yani ters çevirme methodu kullanılmış, bu yüzden uzundan kısaya doğru listeledi

```
renkler = {
  "1.renk": "Gri",
  "2.renk": "Pembe",
  "3.renk": "Mavi"
}
renkler.clear()
print(renkler)
```

```
{ }
```

```
kisiler = {
"1.kişi" : "Sabri" ,
"2.kişi" : "Şükrü" ,
"3.kişi" : "Niyazi"
}
fonksiyon=kisiler.copy()
print(fonksiyon)
```

```
{'1.kişi': 'Sabri', '2.kişi': 'Şükrü', '3.kişi':
'Niyazi'}
```

```
degiskenler = ('1', '2', '3')
degiskenlereEklenecek = 0
fonksiyon = dict.fromkeys(degiskenler,
degiskenlereEklenecek)

print(fonksiyon)
```

```
#dict.fromkeys() : her değişkene 1 değer eklemeyi
sağlar
```

```
sayilar = ('1', '2', '3')
abakus = dict.fromkeys(sayilar)
print(abakus)
```

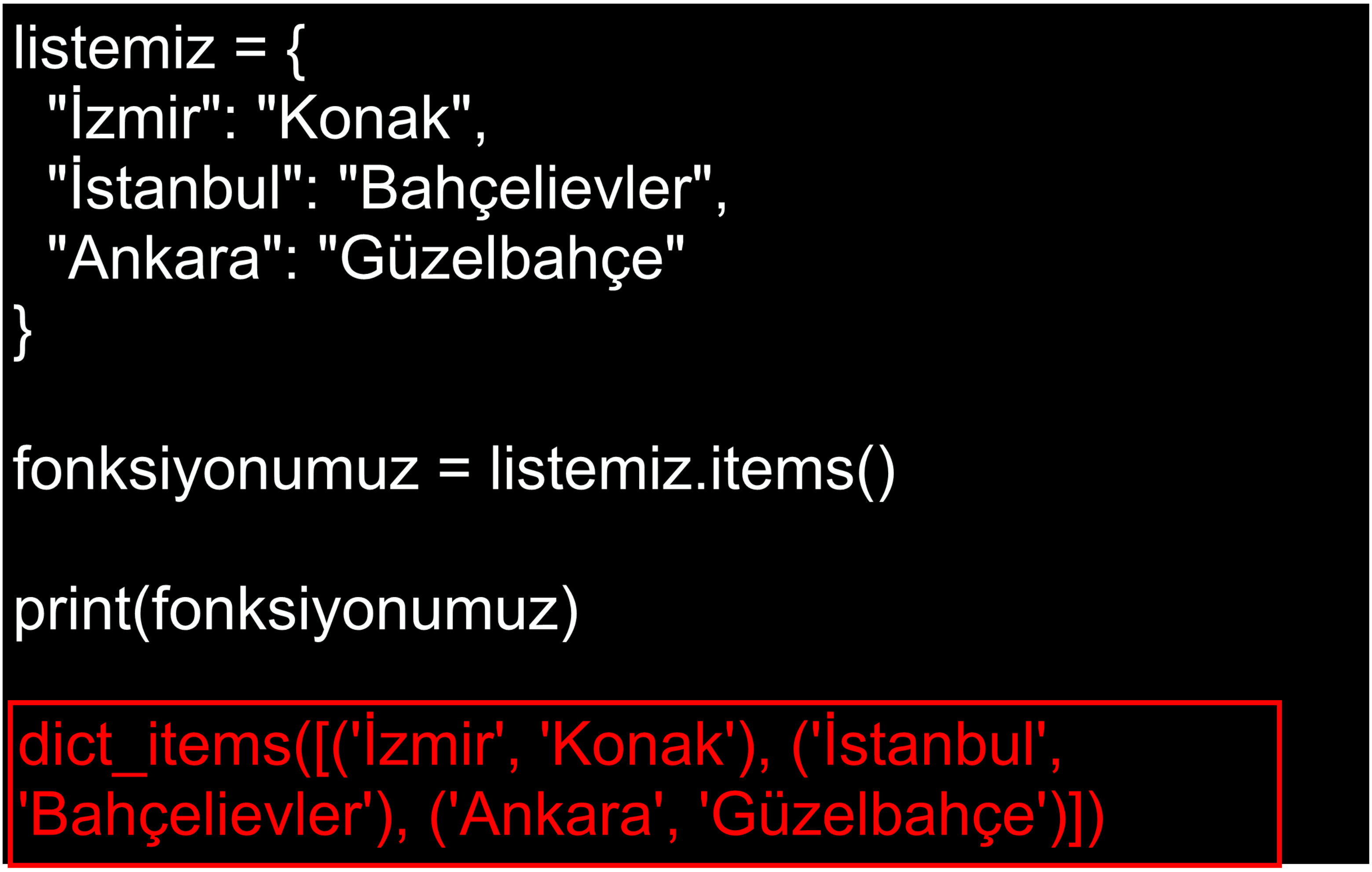
```
{'1': None, '2': None, '3': None}
```

```
#dict.fromkeys(x,y) değerleri alır, sadece sayilar
değişkeni girildiği için onu x varsaydı
```

```
sebzeler = {
  "Sarı": "Limon",
  "Kırmızı": "Greyfurt",
  "Yeşil": "Enginar"
}

fonksiyon = sebzeler.get("Yeşil")

print(fonksiyon)
```

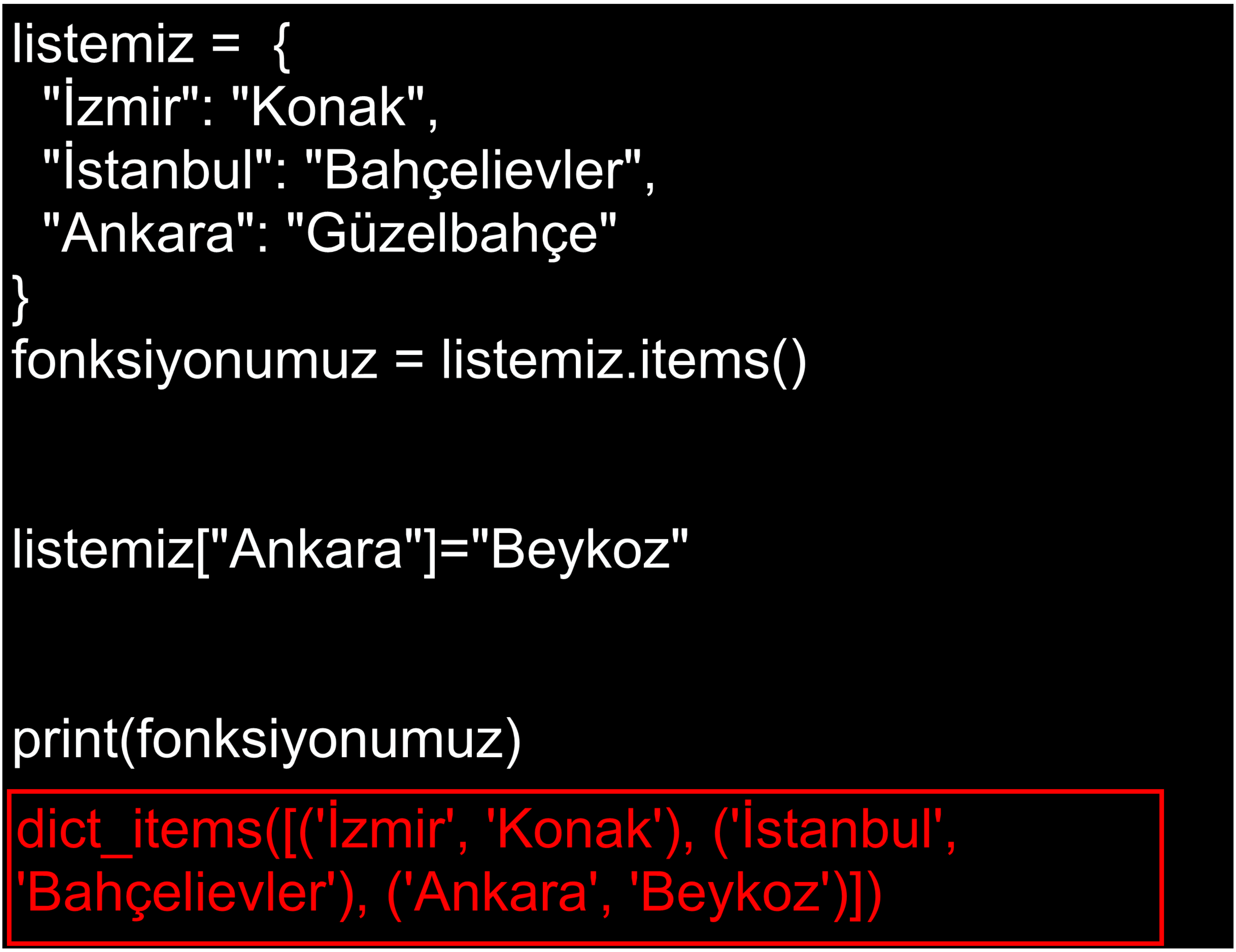
```
Enginar
```

```
#Enginar karşılığı olmasaydı, none değeri
döndürecekti

# ("Yeşil","Karpuz") olsaydı, çıktı olarak Karpuz
alacaktık
```



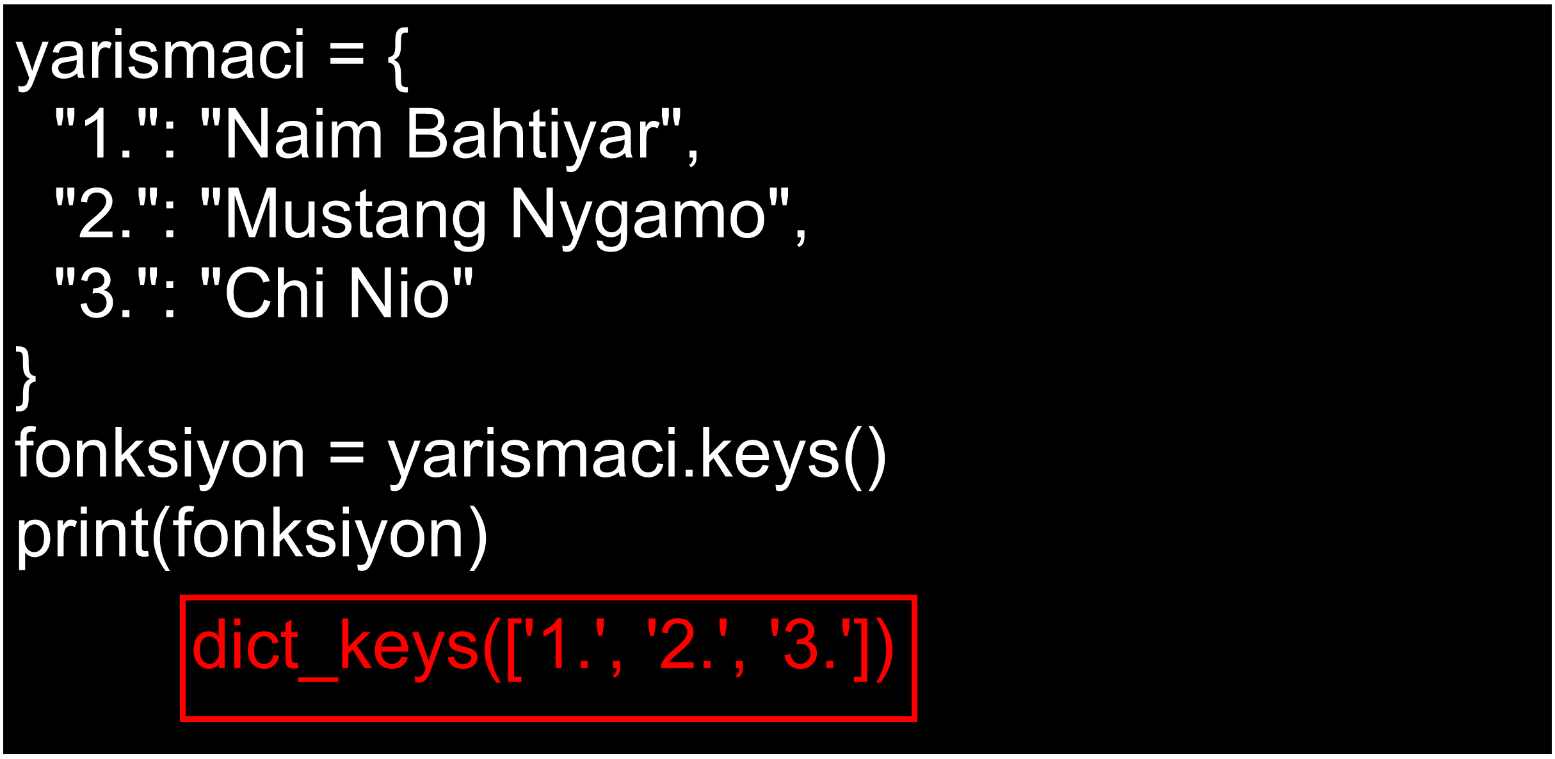
```
listemiz = {
  "İzmir": "Konak",
  "İstanbul": "Bahçelievler",
  "Ankara": "Güzelbahçe"
}

fonksiyonumuz = listemiz.items()

print(fonksiyonumuz)
```

```
dict_items([('İzmir', 'Konak'), ('İstanbul',
'Bahçelievler'), ('Ankara', 'Güzelbahçe')])
```



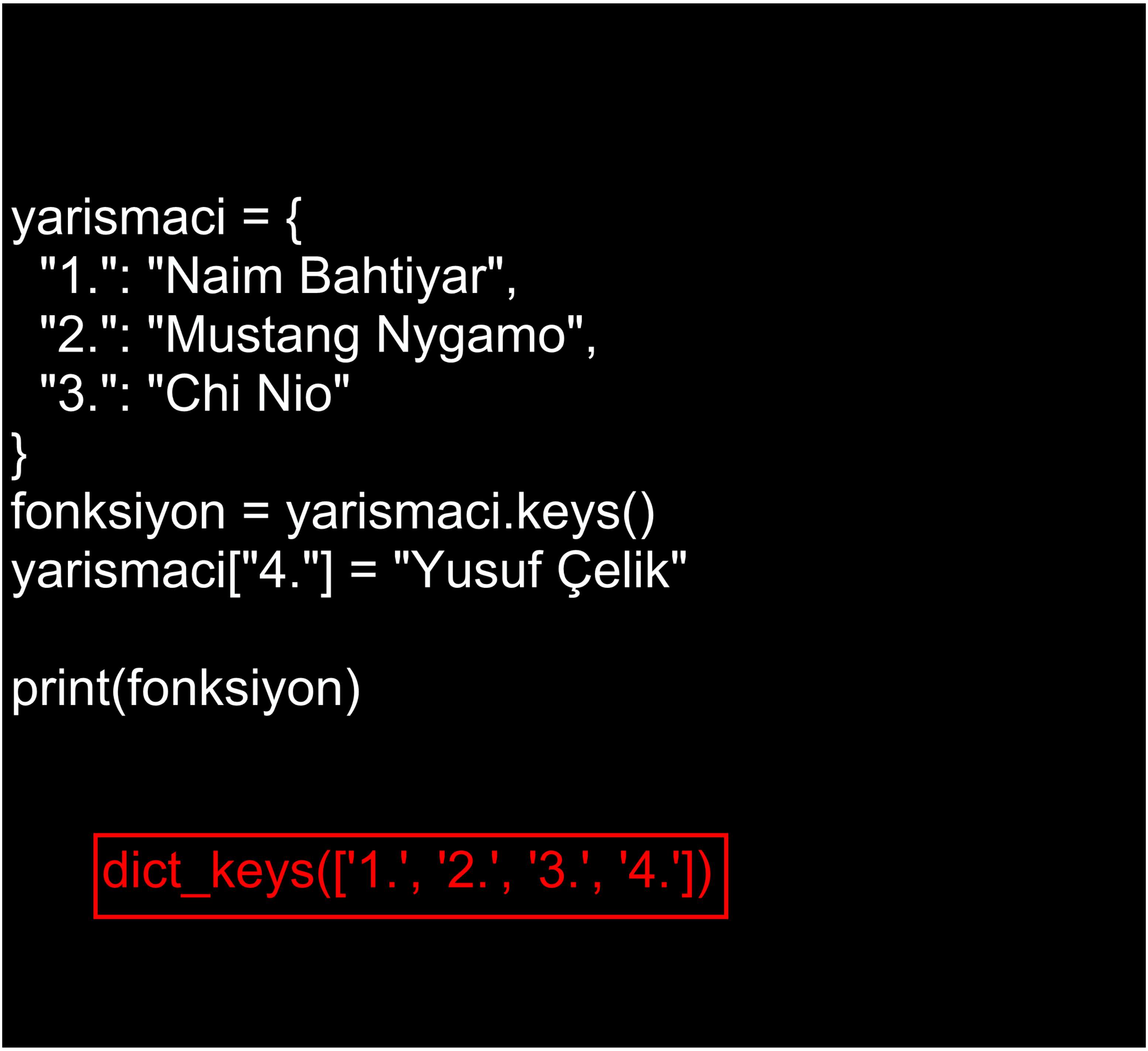
```
listemiz =  {
  "İzmir": "Konak",
  "İstanbul": "Bahçelievler",
  "Ankara": "Güzelbahçe"
}
fonksiyonumuz = listemiz.items()



listemiz["Ankara"]="Beykoz"



print(fonksiyonumuz)
```

```
dict_items([('İzmir', 'Konak'), ('İstanbul',
'Bahçelievler'), ('Ankara', 'Beykoz')])
```



```
yarismaci = {
  "1.": "Naim Bahtiyar",
  "2.": "Mustang Nygamo",
  "3.": "Chi Nio"
}
fonksiyon = yarismaci.keys()
print(fonksiyon)
```

```
dict_keys(['1.', '2.', '3.'])
```



```
yarismaci = {
  "1.": "Naim Bahtiyar",
  "2.": "Mustang Nygamo",
  "3.": "Chi Nio"
}
fonksiyon = yarismaci.keys()
yarismaci["4."] = "Yusuf Çelik"

print(fonksiyon)
```

```
dict_keys(['1.', '2.', '3.', '4.'])
```

```
yarismaci = {
  "1.": "Naim Bahtiyar",
  "2.": "Mustang Nygamo,İlyas Erden",
  "3.": "Chi Nio"
}
yarismaci.pop("2.")
```

```
# pop : kaldırır
# 2. olan yarışmacılar bu fonksiyon ile listeden
çıkartıldı
```

```
print(yarismaci)
```

```
{'1.': 'Naim Bahtiyar', '3.': 'Chi Nio,John Dennys'}
```

```
öğrenciNo = {
  "7-A": 924,
  "9-B": 729,1028,
  "10-C": 1964
}
fonksiyonumuz = öğrenciNo.pop("10-C")

print(fonksiyonumuz)
```

```
1964
```

```
öğrenciNo = {
  "7-A": 924,
  "9-B": 729,
  "10-C": 1964
}
öğrenciNo.popitem()
```

```
# pop = kaldırır,siler
# herhangi bir hedef belirtilmediği için son anahtarı
(10-C'yi) siler
```

```
print(öğrenciNo)
```

```
{'7-A': 924, '9-B': 729}
```

```
arabalar = {
  "araba1": 1995,
  "araba2": 2000,
  "araba3": 2005
}

fonksiyon = arabalar.popitem()
print(fonksiyon)
```

```
('araba3', 2005)
```

```
#üstteki örnekten farklı olarak yaptığımız şey,
fonksiyon kullanmaktı
```

```
odemeler = {
  "Elektrik": 210,
  "Su": 175,
  "Doğalgaz": 290
}

fonksiyonum = odemeler.setdefault("Su", "180")
print(fonksiyonum)
```

```
175
```

```
odemeler = {
  "Elektrik": 210,
  "Su": 175,
  "Doğalgaz": 290
}

fonksiyonum = odemeler.setdefault("İnternet",
180)
print(fonksiyonum)
```

```
180
```

"odemeler" listesinde, "İnternet" öğesi mevcut değil, bu yüzden sadece belirtilen değeri(180) yazdı

```
odemeler = {
  "Elektrik": 210,
  "Su": 175,
  "Doğalgaz": 290
}

odemeler.update({"İnternet":"180"})
print(odemeler)
```

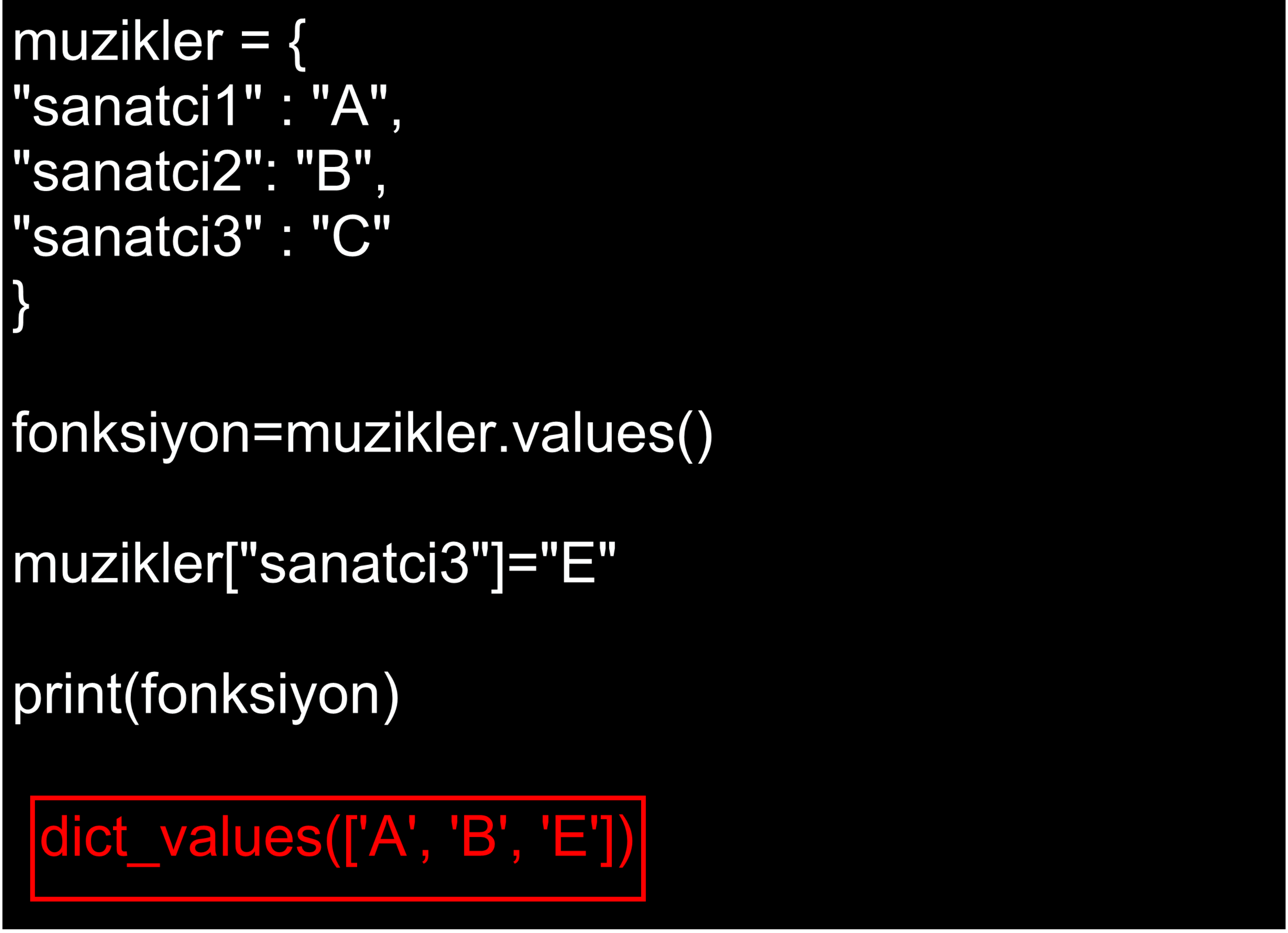
```
{'Elektrik': 210, 'Su': 175, 'Doğalgaz': 290,
'İnternet': '180'}
```

```
# listemize, internet verisini ekledik
```

```
odemeler = {
  "Elektrik": 210,
  "Su": 175,
  "Doğalgaz": 290,
  "İnternet": 180
}

odemeler.values()
print(odemeler)
```

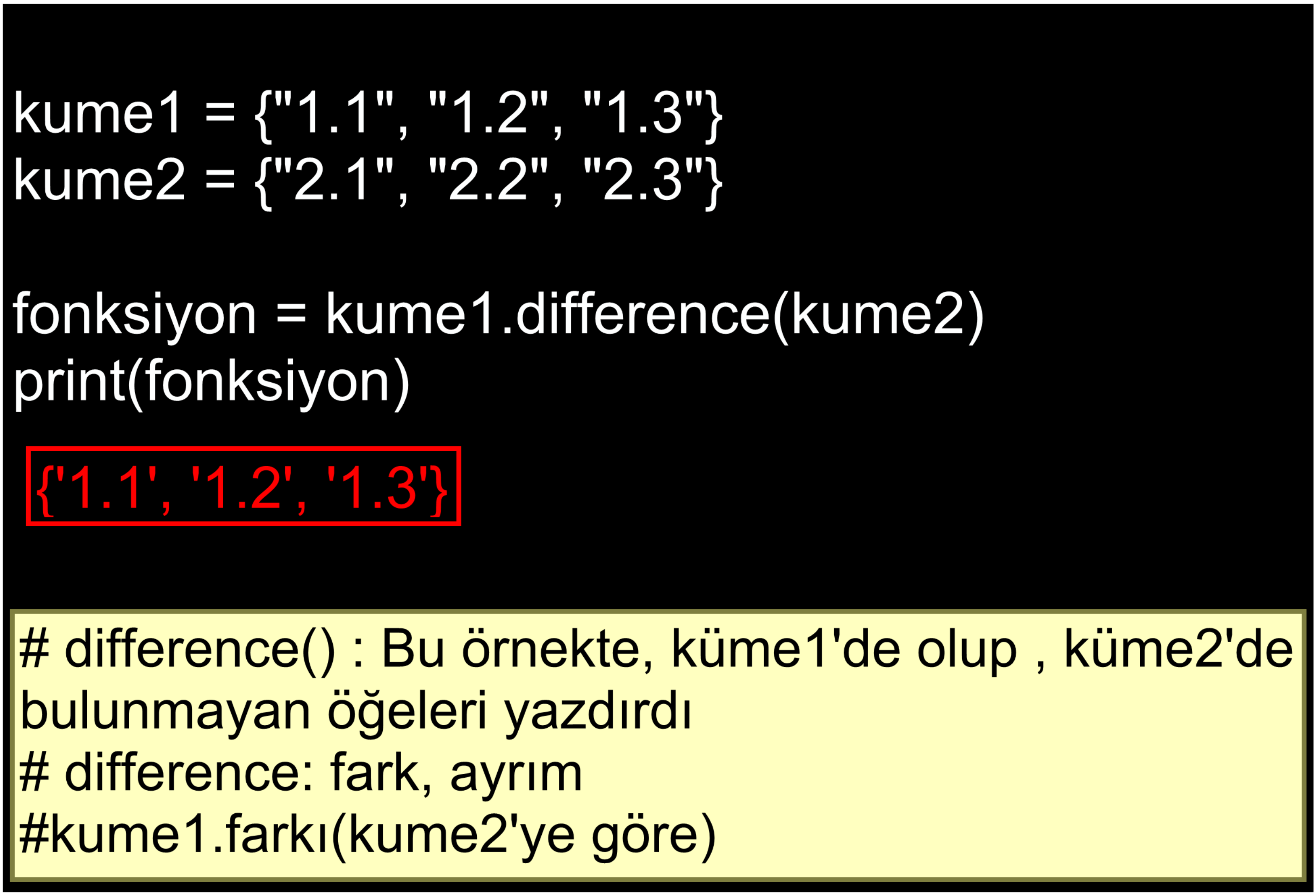
```
{'Elektrik': 210, 'Su': 175, 'Doğalgaz': 290,
'İnternet': 180}
```

```
# listemizdeki değerleri ekrana yazdırdık
```



```
muzikler = {
"sanatci1" : "A",
"sanatci2": "B",
"sanatci3" : "C"
}

fonksiyon=muzikler.values()

muzikler["sanatci3"]="E"

print(fonksiyon)
```

```
dict_values(['A', 'B', 'E'])
```



```
kume1 = {"1.1", "1.2", "1.3"}
kume2 = {"2.1", "2.2", "2.3"}

fonksiyon = kume1.difference(kume2)
print(fonksiyon)
```

```
{'1.1', '1.2', '1.3'}
```

```
# difference() : Bu örnekte, küme1'de olup , küme2'de
bulunmayan öğeleri yazdırdı
# difference: fark, ayrım
#kume1.farkı(kume2'ye göre)
```

```
kume1 = {"1.1", "a", "b"}
kume2 = {"2.1", "a", "b"}

kesisimleri = kume2.difference(kume1)
print(kesisimleri)
```

```
{'2.1'}
```

```
kume1 = {"1", "10", "100"}
kume2 = {"50", "5", "10"}

kume1.difference_update(kume2)

print(kume1)



#küme1.kesişim(küme2)
```

```
#kesişim=farklı olan yani küme1'de olup ,küme2'de
olmayan

#farklılık söz konusu olduğundan, 2 kümede de
bulunan 10 değeri gider
```

```
{'1', '100'}
```

```
liste = {"yoğut", "ekmek", "yumurta"}
liste.discard("ekmek")
print(liste)
```

```
{'yumurta', 'yoğut'}
```

```
#discard() : listeden,belirlenen öğeyi siler
```

```
kume1 = {"0", "1", "2"}
kume2 = {"1", "3", "5"}

birlesim = kume1.intersection(kume2)

print(birlesim)
```

```
{'1'}
```

```
#intersection() : kümelerin ortak elemanlarını getirir
```

```
kume1 = {"0","1","2"}
kume2 = {"7","2","4"}
kume3 = {"2","5","8"}
birlesim = kume1.intersection(kume2, kume3)
print(birlesim)
```

```
{'2'}
```

```
# 3 kümede de ortak olan eleman 2'dir
```

```
kume1 = {"futbol", "basketbol", "tenis"}
kume2 = {"araba", "kamyonet", "bisiklet"}

fonksiyon = kume1.isdisjoint(kume2)

print(fonksiyon)
```

```
#kume1'in elemanları, küme2'de hiçbir öğe ile aynı
değilse True döndürür , aynı olan var ise False
döndürür
```

```
sinif1 = {"can", "okan", "canan"}
sinif2 = {"can", "okan", "canan", "abdurrezzak"}

fonksiyon = sinif1.issubset(sinif2)

print(fonksiyon)
```

```
True
```

```
# issubset() : altküme



# sinif1.elemanlarını(sinif2'de ara)
#var ise True , yok ise False döndür
```

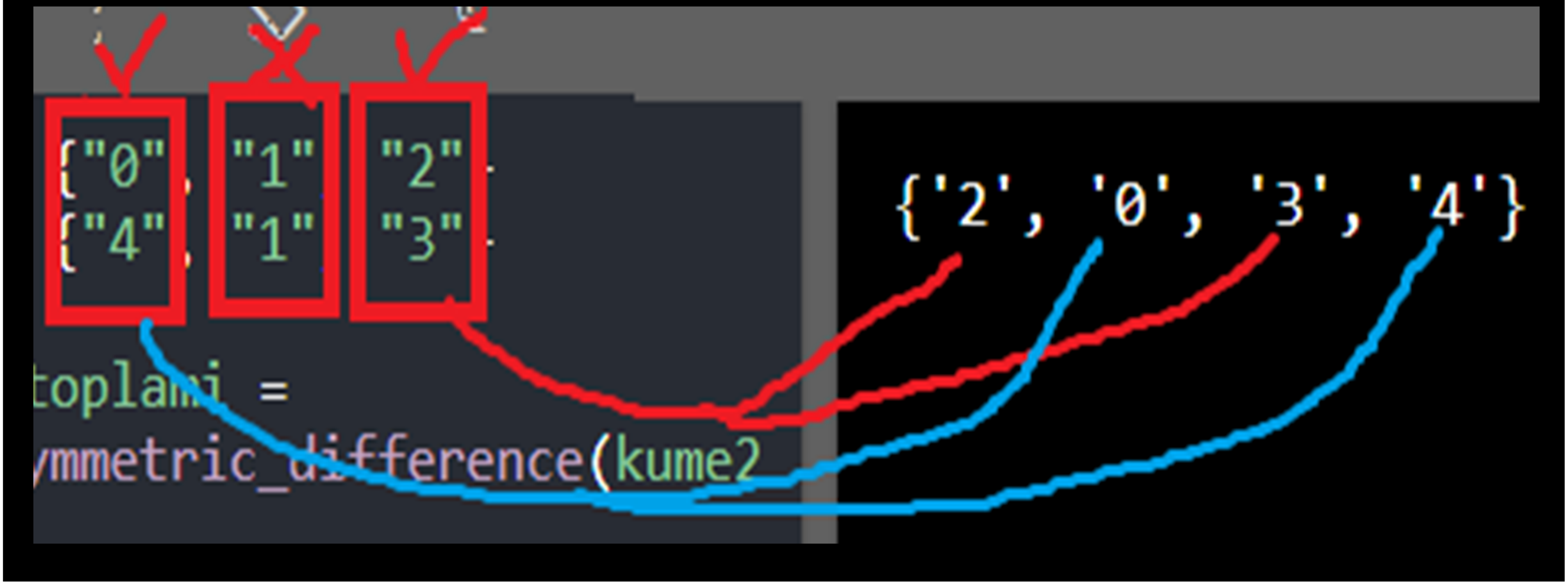
```
x_kumesi = {"a", "b", "c", "d", "e", "f"}
y_kumesi = {"b", "c", "d"}

fonksiyon = x_kumesi.issuperset(y_kumesi)
```

```
# y kümesindeki öğeler, x kümesinde bulunuyorsa
True döndürür, bulunmuyorsa False döndürür
```

```
print(fonksiyon)    True
```

```
kume1 = {"0", "1", "2"}
kume2 = {"4", "1", "3"}

kumelertoplami =
kume1.symmetric_difference(kume2)

print(kumelertoplami)
```

```
{'2', '0', '3', '4'}
```



```
{"0", "1", "2"
{"4", "1", "3"

toplami =
ymmetric_difference(kume2
```

```
{'2', '0', '3', '4'}
```

```
kume1 = {"a", "b", "c"}
kume2 = {"a", "2", "3"}

kume1.symmetric_difference_update(kume2)

print(kume1)
```

```
{'b', '2', 'c', '3'}
```

```
liste1 = {"öğe1.1", "öğe1.2", "öğe1.3"}
liste2 = {"öğe1.2", "öğe1.1", "öğe2.1"}
fonksiyon = liste1.union(liste2)
print(fonksiyon)
```

```
{'öğe1.1', 'öğe1.2', 'öğe1.3', 'öğe2.1'}
```

```
#union : birleşme
#union() : kümelerin her öğesini birer kere alır
```

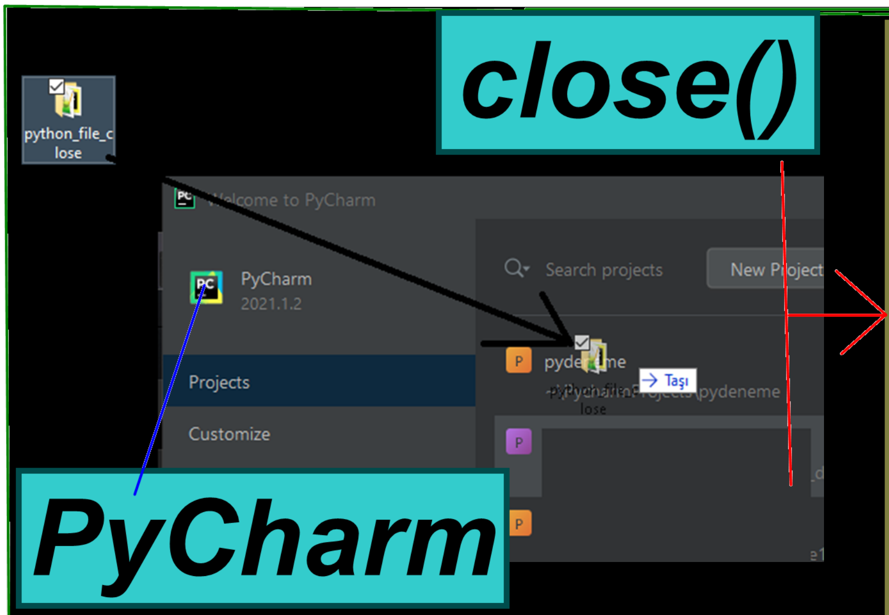
```
kume1 = {"a", "b", "c"}
kume2 = {"f", "d", "a"}
kume3 = {"c", "d", "e"}

fonksiyonum = kume1.union(kume2, kume3)

print(fonksiyonum)
```

```
{'e', 'a', 'c', 'f', 'b', 'd'}
```

```
#Bir önceki örnekten farklı olarak 3 tane liste/küme
kullandık ve union() ile listelerin öğelerini birer tane
olmak üzere çeker
```

```
liste_1 = {"Osman", "Kemal", "Faruk"}
liste_2 = {"Dilek", "Melis", "Aysu"}

liste_1.update(liste_2)
print(liste_1)
```

```
{'Aysu', 'Dilek', 'Melis', 'Kemal', 'Faruk', 'Osman'}
```


close()
python_file_close
Welcome to PyCharm
PyCharm
2021.1.2
Search projects
New Project
Projects
Customize
pydeneme
Taşı
PyCharm

->Masaüstüne bir klasör oluşturduk ve adını python_file_close yaptık , daha sonra editörümüze sürükleyip bıraktık. Bizim için yeni proje oluşturdu .

-> Masaüstümüzdeki python_file_close klasörüne çift tıklayıp açtık ve daha sonra içine metin_belgemiz.txt adında metin belgesi oluşturduk ve metin belgesinin içine girip "Bu metin belgesinin icerigidir" yazdık ve Ctrl+S yapıp kaydettik ve kapattık

```
fonksiyonum = open("metin_belgemiz.txt", "r")
print(fonksiyonum.read())
fonksiyonum.close()
```

```
Bu metin belgesinin icerigidir
```

```
fonksiyonum = open("metin_belgemiz.txt", "r")

print(fonksiyonum.fileno())
```

```
3
```

fileno() : İşletim sisteminin bakış açısından,
akışı temsil eden bir sayı döndürür

```
fonks = open("metin_belgemiz.txt", "a")

fonks.write("Simdi dosyada 1 satir var")

fonks.flush()

fonks.write("...ve yeni bir satir daha yazildi")
```

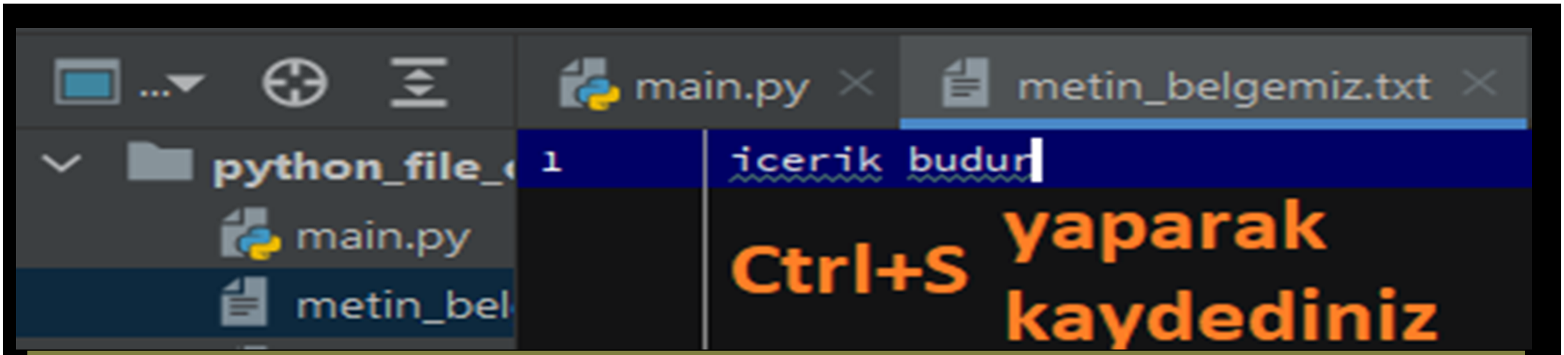
```
# çıktısı (metin_belgemiz1 üzerine yazar) : Simdi
dosyada 1 satir var...ve yeni bir satir daha yazildi
```

flush() : Dahili arabelleği temizler

```
fonksiyon = open("metin_belgemiz.txt", "r")
print(fonksiyon.isatty())
```

```
False
```

isatty() : Dosya akışının etkileşimli olup olmadığını
döndürür



```
icerik budur
```

Ctrl+S yaparak kaydediniz

metin_belgemiz.txt dosyamızı açıp "icerik budur" yazalım ve Ctrl+S kombinasyonuyla kaydedelim , ardından main.py dosyamıza geçelim ve kodlarımızı yazmaya başlayalım

```
fonksiyon = open("metin_belgemiz.txt","r")
print(fonksiyon.read())
```

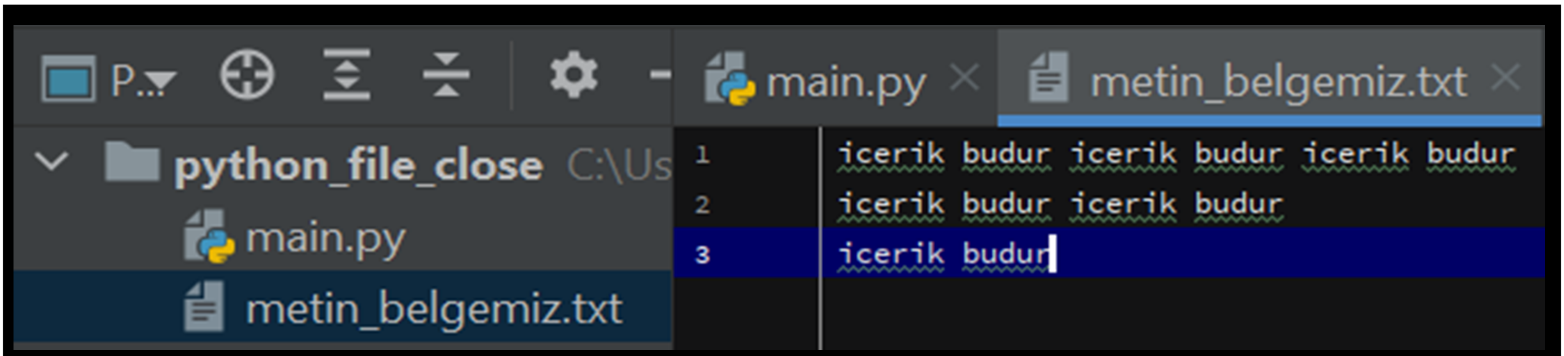
```
icerik budur
```

```
# read() : dosyanın içeriğini okur

# read(9) yazsaydık, metin_belgemiz.txt içerisinden
ilk 9 indexi getirirdi yani çıktısı "icerik bu" olurdu
```

```
fonksiyonumuz = open("metin_belgemiz.txt", "r")
print(fonksiyonumuz.readable())
```

```
True
```

```
# readable() : Dosya içeriğinin, okunup
okunamayacağını döndürür
```


P..
python_file_close C:\Us
main.py
metin_belgemiz.txt
main.py
metin_belgemiz.txt

```
icerik budur icerik budur icerik budur
icerik budur icerik budur
icerik budur
```

metin_belgemiz.txt 'ye bu şekilde 3 satır yaptık, şimdi ise main.py dosyamıza geçelim ve kodlarımızı yazalım

```
fonksiyon = open("metin_belgemiz.txt", "r")
print(fonksiyon.readline())
```

```
icerik budur icerik budur icerik budur
```

```
#readline() : "metin_belgemiz.txt" dosyasının ilk
satırını okur

#Hem 1. hem de 2. satırı okuması için 2 kere
print(fonksiyon.readline())
print(fonksiyon.readline())
yazılması gerekir

#print(fonksiyon.readline(5)) şeklinde yazmamız, ilk 5
indexi yazdırır, çıktı olarak "iceri" 'yi alırız
```

```
fonksiyon = open("metin_belgemiz.txt", "r")
print(fonksiyon.readlines())
```

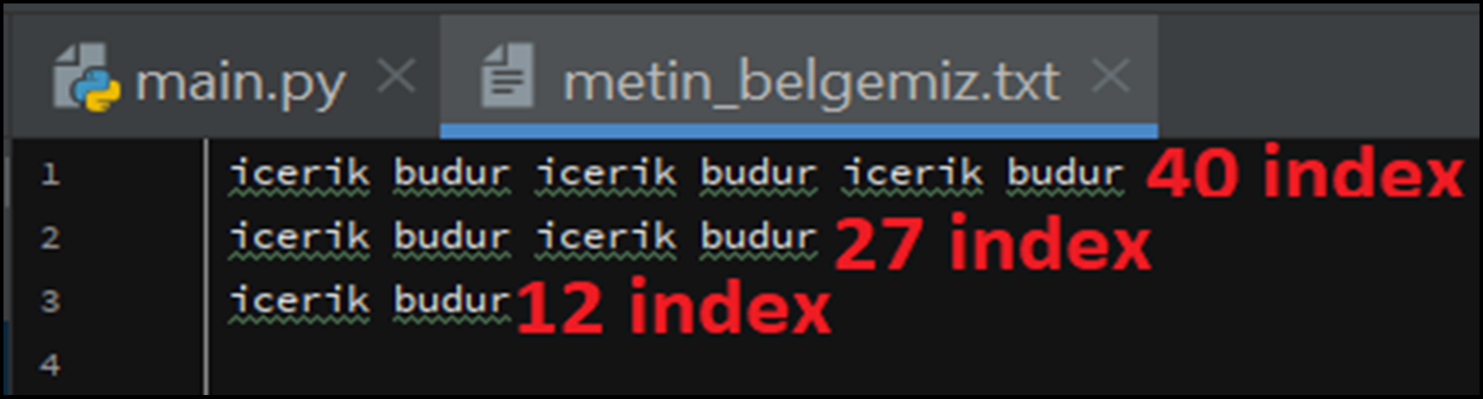
```
['icerik budur icerik budur icerik budur\n', 'icerik budur icerik budur\n', 'icerik budur']
```

```
# readlines() : Bütün satırları okur,alt satıra geçildiğini \n ile belirtir
```

```
fonksiyonumuz = open("metin_belgemiz.txt", "r")
print(fonksiyonumuz.readlines(7))
```

```
#7 byte = 7*8 = 56 harf , yani 56 harften fazlaysa diğer satıra geçme demiş olduk
```

```
['icerik budur icerik budur icerik budur\n']
```



```
icerik budur icerik budur icerik budur
icerik budur icerik budur
icerik budur

```

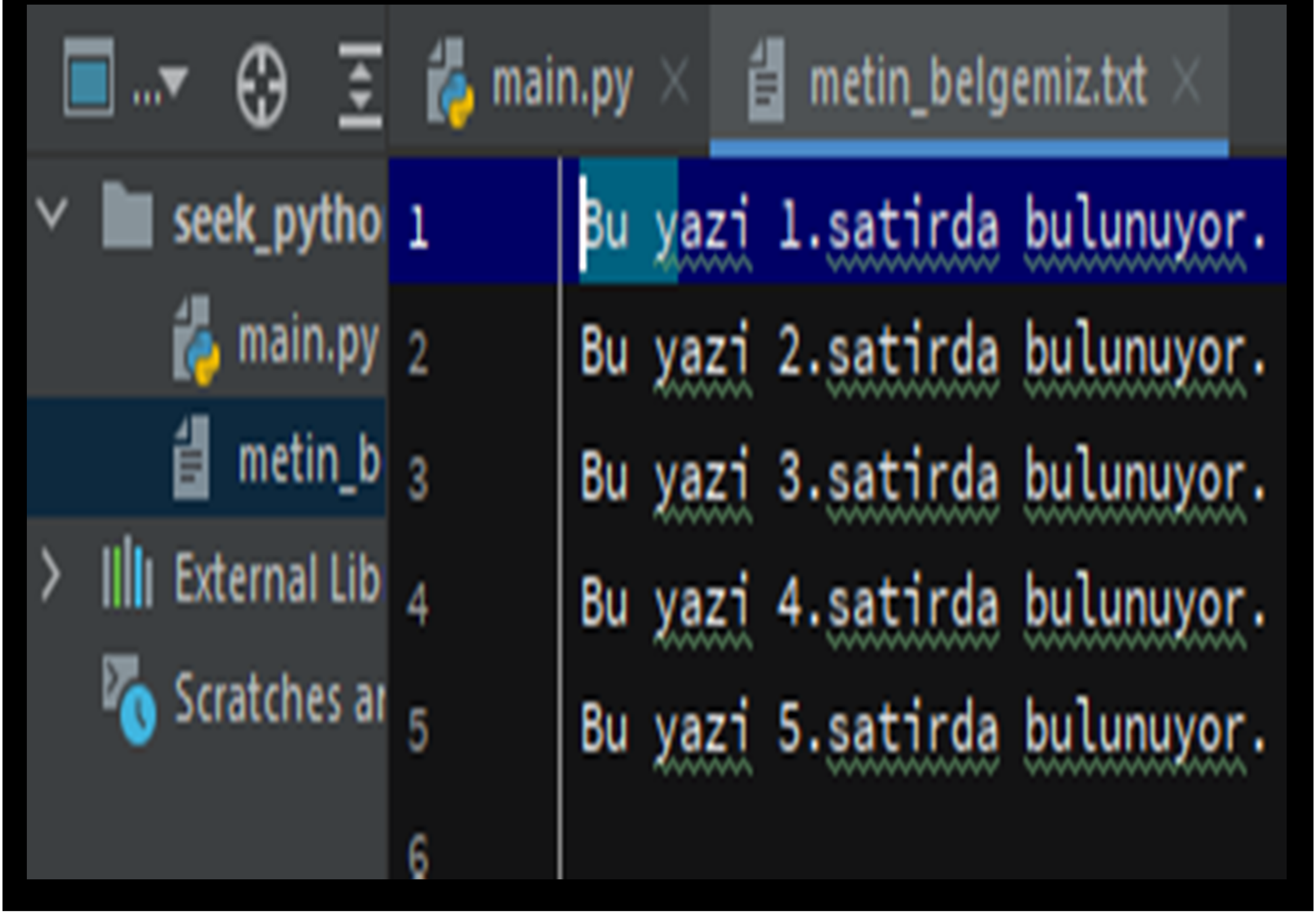
1 byte = 8 harf

8 index = 8 karakter

```
fonksiyon = open("metin_belgemiz.txt", "r")
fonksiyon.seek(5)
#Başlangıçtaki satıra gitmesini istedik

#Satırdaki 5. bayt'a gider ve oraya kadarki kısmı
#keser ve kalan kısmı ekrana yazdırır

print(fonksiyon.readline())
```

```
zi 1.satirda bulunuyor.
```

```
fonksiyon.close()
```


main.py
metin_belgemiz.txt
seek_pytho
main.py
metin_b
External Lib
Scratches ar

```
Bu yazi 1.satirda bulunuyor.
Bu yazi 2.satirda bulunuyor.
Bu yazi 3.satirda bulunuyor.
Bu yazi 4.satirda bulunuyor.
Bu yazi 5.satirda bulunuyor.
```

```python
fonksiyon = open("metin_belgemiz.txt","r")
print(fonksiyon.seek(4))
```

```
4
```

```python
# seek() = Yeni konumu döndürdü
# seek() = Dosya konumunu değiştirir
```

```python
fonksiyon = open("metin_belgemiz.txt","r")
print(fonksiyon.seekable())
```

```
True
```

```python
# seekable : aranabilir
# seekable() : Dosyanın dosya konumunu
değiştirmemize izin verip vermediğini döndürür
```

```python
fonksiyon = open("metin_belgemiz.txt","r")
print(fonksiyon.tell())
```

```
0
```

```python
# tell() : Geçerli dosya konumunu bulur
```

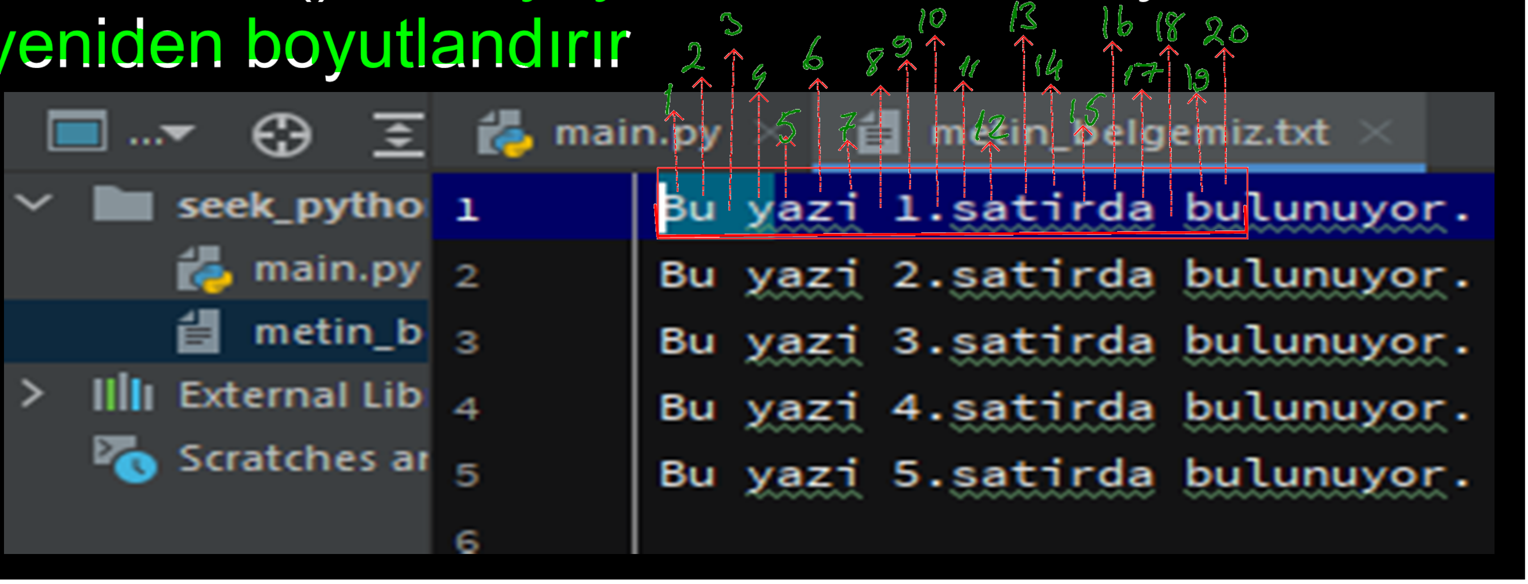
```
fonksiyon = open("metin_belgemiz.txt","r")
print(fonksiyon.readline())
# İlk satırı okuduktan sonra mevcut dosya konumunu
döndürür

print(fonksiyon.tell())
```

```
Bu yazi 1.satirda bulunuyor.

        30
```

```
fonksiyon = open("metin_belgemiz.txt","a")
fonksiyon.truncate(20)
fonksiyon.close()

fonksiyon=open("metin_belgemiz.txt","r")
print(fonksiyon.read())
```

```
Bu yazi 1.satirda bu
```

```
# truncate() : Dosyayı, belirtilen bir boyuta
yeniden boyutlandırır
```



```
Bu yazi 1.satirda bulunuyor.
Bu yazi 2.satirda bulunuyor.
Bu yazi 3.satirda bulunuyor.
Bu yazi 4.satirda bulunuyor.
Bu yazi 5.satirda bulunuyor.
```

main.py
metin_belgemiz.txt
seek_pytho
main.py
metin_b
External Lib
Scratches ar
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

```
fonksiyon=open("metin_belgemiz.txt","a")
print(fonksiyon.writable())
```

```
True
```

```
# writable() : Dosyanın, yazdırılıp yazdırılamayacağını
gösterir
```

```
fonksiyon = open("metin_belgemiz.txt","a")

fonksiyon.truncate(0)
# Bu kod sayesinde, program her çalıştırıldığında
# write komutu sadece 1 kere yazdıracak

fonksiyon.write("\nYazdırma fonksiyonunu
kullanıyoruz")

fonksiyon.close()

fonksiyon=open("metin_belgemiz.txt","r")

print(fonksiyon.read())
```

```
Yazdırma fonksiyonunu kullanıyoruz
```

```
# write() : Belirtilen dizeyi, dosyaya yazar
```

```
belge=open("metin_belgemiz.txt","a")
belge.truncate(0)
belge.writelines(["1.kısım", " 2.kısım", " 3.kısım"])
belge.close()
belge = open("metin_belgemiz.txt","r")

print(belge.read())
```

```
1.kısım 2.kısım 3.kısım
```

writelines() : Dosyaya, dizeleri listesini halinde yazar

```
belge=open("metin_belgemiz.txt","a")
belge.truncate(0)
belge.writelines(["1.kısım", "\n2.kısım",
"\n3.kısım"])
belge.close()
belge = open("metin_belgemiz.txt","r")
print(belge.read())
```

```
1.kısım
2.kısım
3.kısım
```

```
degisken = (5>3 and 5<10)
print(degisken)
```

```
True
```

```
#Her 2 olasılık da doğru ise True döndürür
```

```
if 5>3 and 5<10:
      print("2 Olasılık da doğru")
else:
      print("2 Olasılık da doğru değil")
```

```
#and :ve operatörüdür
```

```
2 Olasılık da doğru
```

```
import calendar as c

print(c.month_name[2])
```

```
February
```

```
#Takma ad oluşturmak için as anahtar sözcüğü
kullanılır.

#Yukarıdaki örnekte, takvim modülünü içe aktarırken
bir takma ad (c) oluşturuyoruz,
artık takvim yerine c kullanarak, takvim modülüne
başvurabiliyoruz.

#as : gibi
```

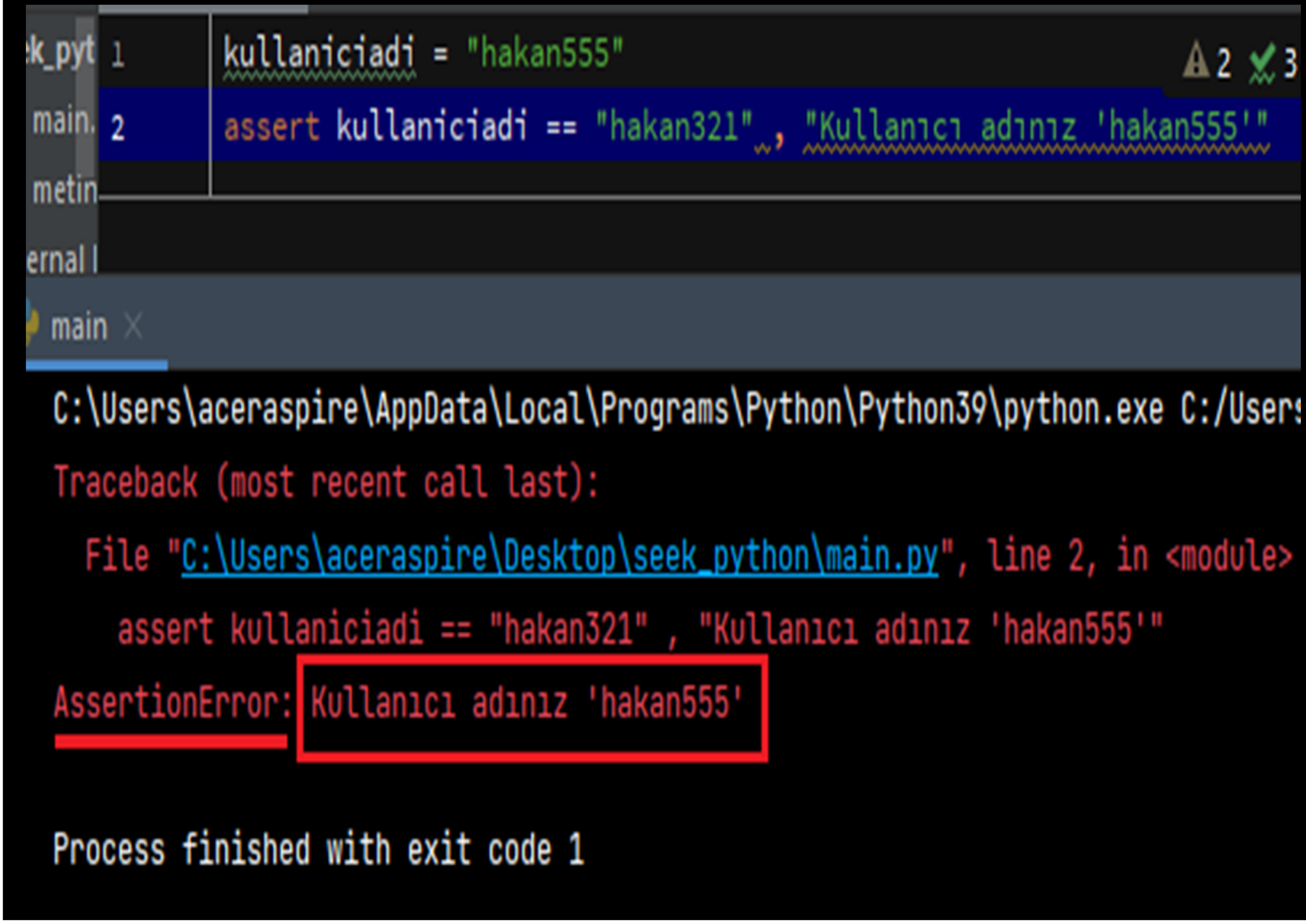
```
sifrem = "123"
assert sifrem == "123"
```

Bu durumdayken, True değerini döndürürse hiçbir şey yazdırmaz

False değerini döndürürse, program hata verir

#assert : iddia etmek, savunmak

```
kullaniciadi = "hakan555"
assert kullaniciadi == "hakan321" , "Kullanıcı adınız 'hakan555'"
```



```
kullaniciadi = "hakan555"
assert kullaniciadi == "hakan321" , "Kullanıcı adınız 'hakan555'"
```

```
C:\Users\aceraspire\AppData\Local\Programs\Python\Python39\python.exe C:/Users
Traceback (most recent call last):
  File "C:\Users\aceraspire\Desktop\seek_python\main.py", line 2, in <module>
    assert kullaniciadi == "hakan321" , "Kullanıcı adınız 'hakan555'"
AssertionError: Kullanıcı adınız 'hakan555'

Process finished with exit code 1
```

```
for dongumuz in range(9):

    if dongumuz>3:
        break
    print(dongumuz)
```

```
0
1
2
3
```

```
#break : mola, kırmak
#break() : Bir döngüden çıkmak için kullanılır

#for dongumuz in range(9):
#Burada for döngüsünü kullandık, döngünün
adına dongumuz dedik ; range aralık demek,
dongumuzu 0-9 aralığında almasını istedik

# if dongumuz>3:
#Burada, dongumuz 3'ten büyük olduğu anda
aşağıdaki adımları uygulasın, dedik

#break
#Burada, dongumuz 3'ten büyük olduğu anda
döngüden çıkmasını istedik
```

```
sayimiz = 1

while sayimiz<9:
    print(sayimiz)
    if sayimiz==3:
        break

    sayimiz += 1
```

```
1
2
3
```

```
class Kisi:
    ad="Metin"
    yas=20

#Kisi adında bir sınıf oluşturduk
```

```
class Kisi:
    ad="Metin"
    yas=20

kisiGetir = Kisi()
print(kisiGetir.ad)
```

```
Metin
```

```
#class : Sınıf tanımlamak için kullanılır
```

```
for dongumuz in range(9):

    if dongumuz == 3:
        continue

    print(dongumuz)
```

```
0
1
2
4
5
6
7
8
```

#continue() : dongumuz değişkeni 3 ise eşitliği atlasın, bir sonraki eşitliğe kadar devam etsin istedik

```
sayi = 0

while sayi<5:
    sayi += 1 #sayıyı 1 artır ve tekrar sayıya eşitle

    if sayi == 3:
        continue

    print(sayi)
```

```
1
2
4
5
```

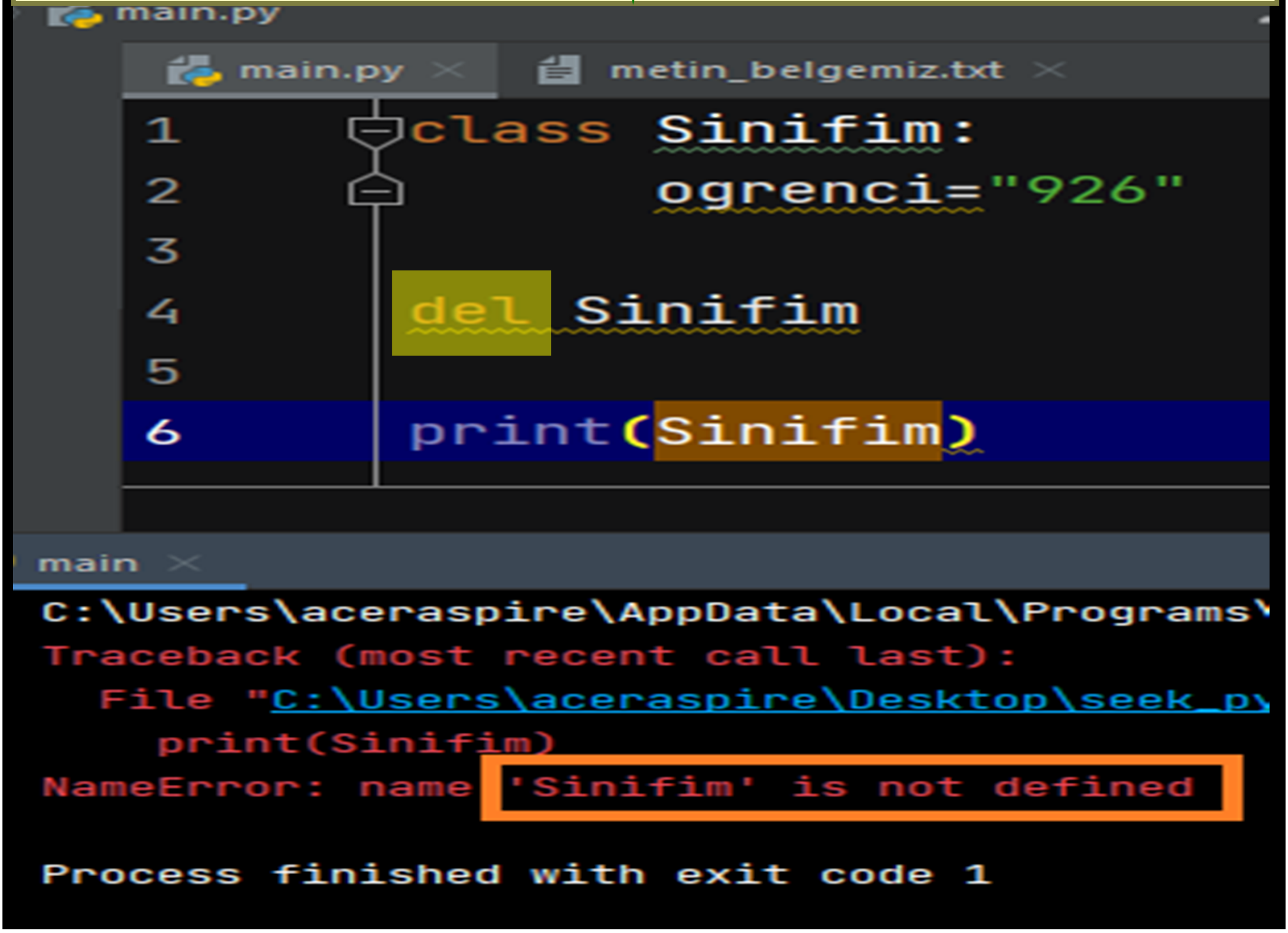
#Devam anahtar sözcüğünü,
while döngüsünde kullandık

#sayi = 3 olduğunda geçmesini istedik

```
def fonksiyonum():
    print("Fonksiyon çalıştığında görünecek")

fonksiyonum()
Fonksiyon çalıştığında görünecek
```

Burada def , bir fonksiyon tanımlamak için kullanıldı

Del anahtar sözcüğü nesneleri silmek için kullanılır. Python'da her şey bir nesnedir, bu nedenle del anahtar sözcüğü değişkenleri, listeleri veya bir listenin bölümlerini vb. silmek için de kullanılabilir.



```
class Sinifim:
        ogrenci="926"

del Sinifim

print(Sinifim)
```

```
C:\Users\aceraspire\AppData\Local\Programs
Traceback (most recent call last):
  File "C:\Users\aceraspire\Desktop\seek_py
    print(Sinifim)
NameError: name 'Sinifim' is not defined

Process finished with exit code 1
```

```
degisken = "Merhaba"
del degisken
print(degisken)
```

```
NameError: name 'degisken' is not defined
```

```
#Gördüğünüz gibi ,değişken silindiği için
bulunamadı hatası veriyor
```

```
liste = ["degisken1","degisken2","degisken3"]

del liste[0]

print(liste)
```

```
['degisken2', 'degisken3']
```

```
# Listedeki ilk öğeyi sildi
```

```python
for sayi in range(-5,5):

    if sayi>0:
        print("Sayı pozitif")
    elif sayi==0:
        print("Sayı 0'a eşit")
    else:
        print("Sayı Negatif")
```

```
Sayı Negatif
Sayı Negatif
Sayı Negatif
Sayı Negatif
Sayı Negatif
Sayı 0'a eşit
Sayı pozitif
Sayı pozitif
Sayı pozitif
Sayı pozitif
```

```
#-5 değeri negatif , -4 değeri negatif ..... 0 .... şeklinde ilerliyor
#elif = else if ile aynı yani else if'in kısaltılmış hali
# if (koşul sağlıyor mu?)
# else if (önceki koşul uymadı peki bu koşul sağlıyor mu?)
```

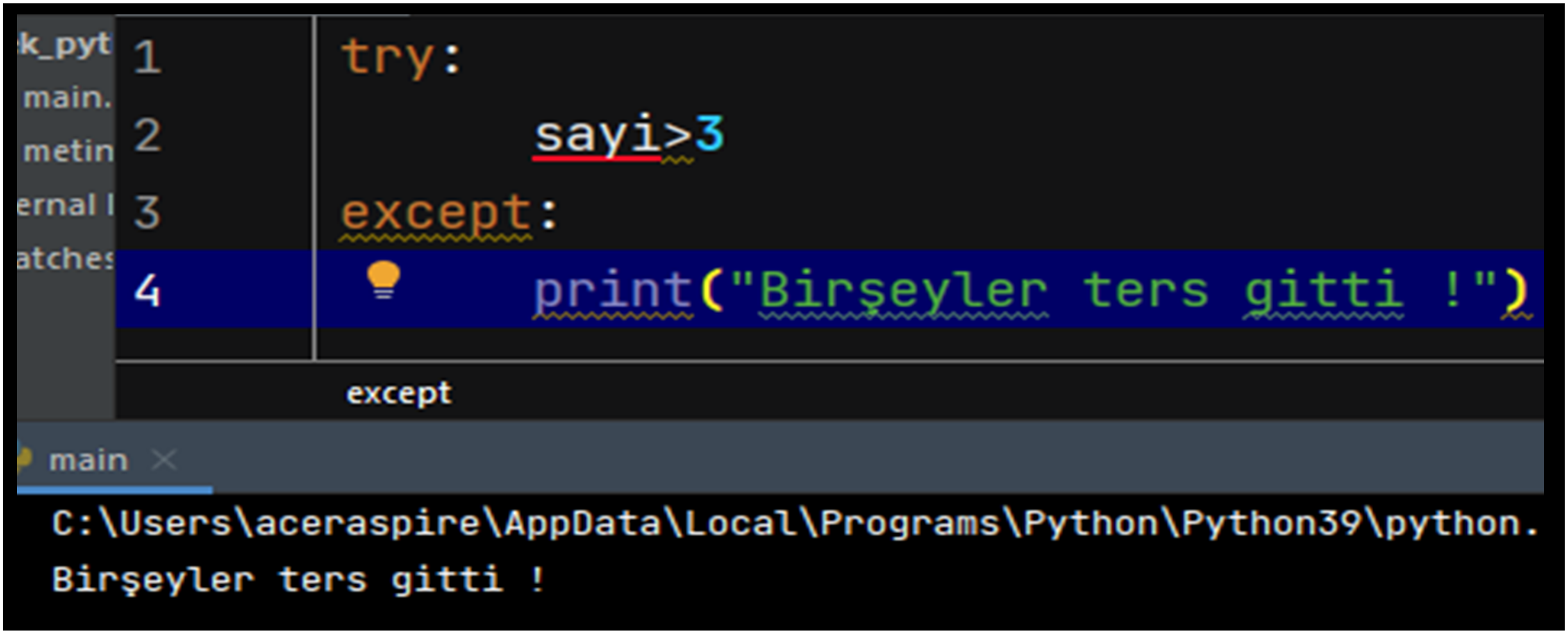
```
sayi = 2

if sayi>3: #sayı 3'ten büyükse
    print("Sayı 3'ten büyük") #bunları yap

else: #sayı 3'ten büyük değilse
    print("Sayı 3'ten büyük değil") #bunları yap
```

```
Sayı 3'ten büyük değil
```

```
sayimiz = 12

try:
    sayimiz>15

except:
    print("Hatalı giriş")

else:
    print("try komutu,hatasız çalıştırıldı")
```

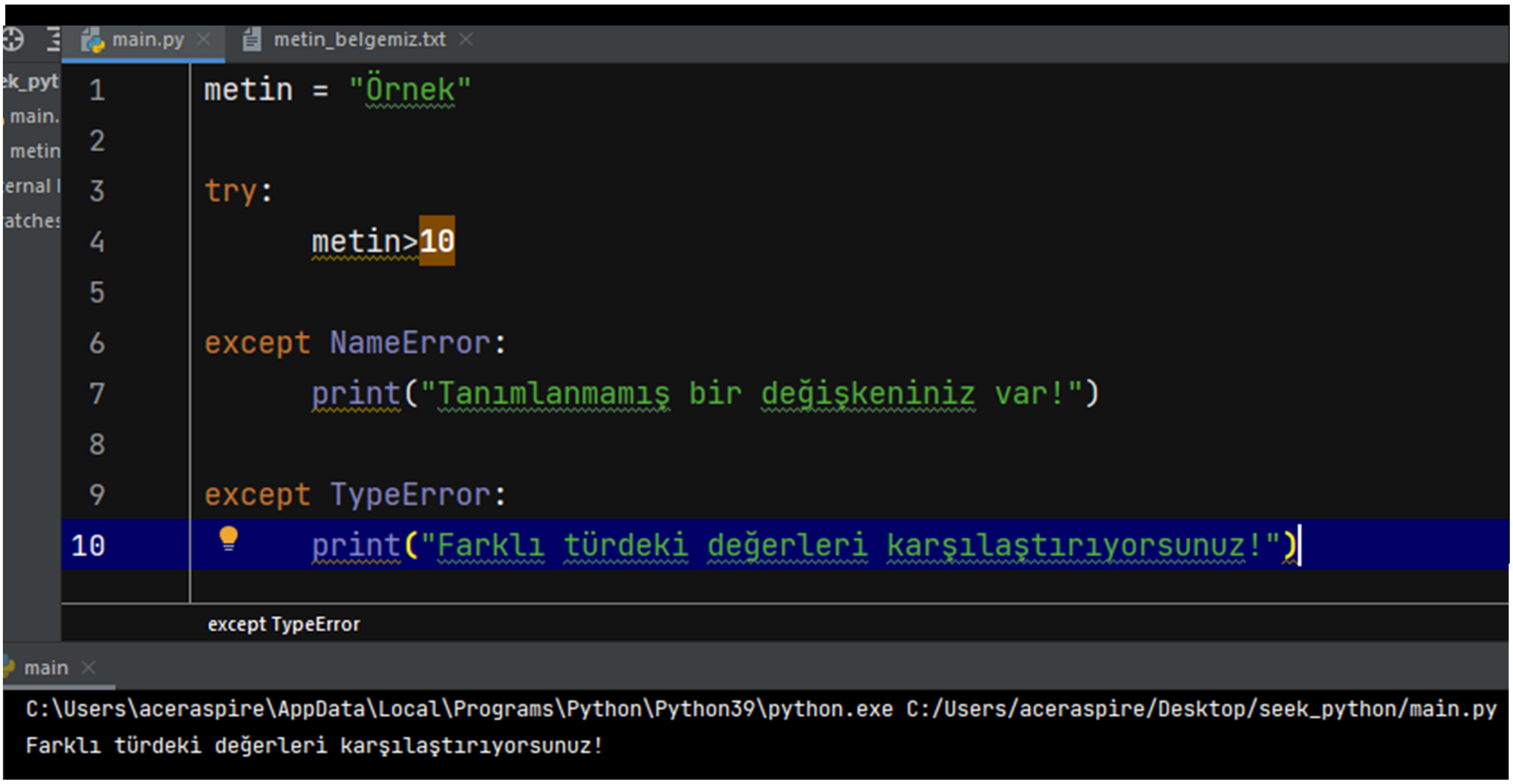
```
try komutu,hatasız çalıştırıldı
```

```
#try:denemek,sınamak
#except:dışında,başka,hariç tutmak,ayrı tutmak
#except() : İstisnalarla birlikte kullanılır, bir istisna
oluştuğunda ne yapılmalı
```


k_pyt
main.
metin
ernal l
atches
except
main

```
try:
        sayi>3
except:
        print("Birşeyler ters gitti !")
```

```
C:\Users\aceraspire\AppData\Local\Programs\Python\Python39\python.
Birşeyler ters gitti !
```


main.py
metin_belgemiz.txt
k_pyt
main.
metin
ernal l
atches
except TypeError
main

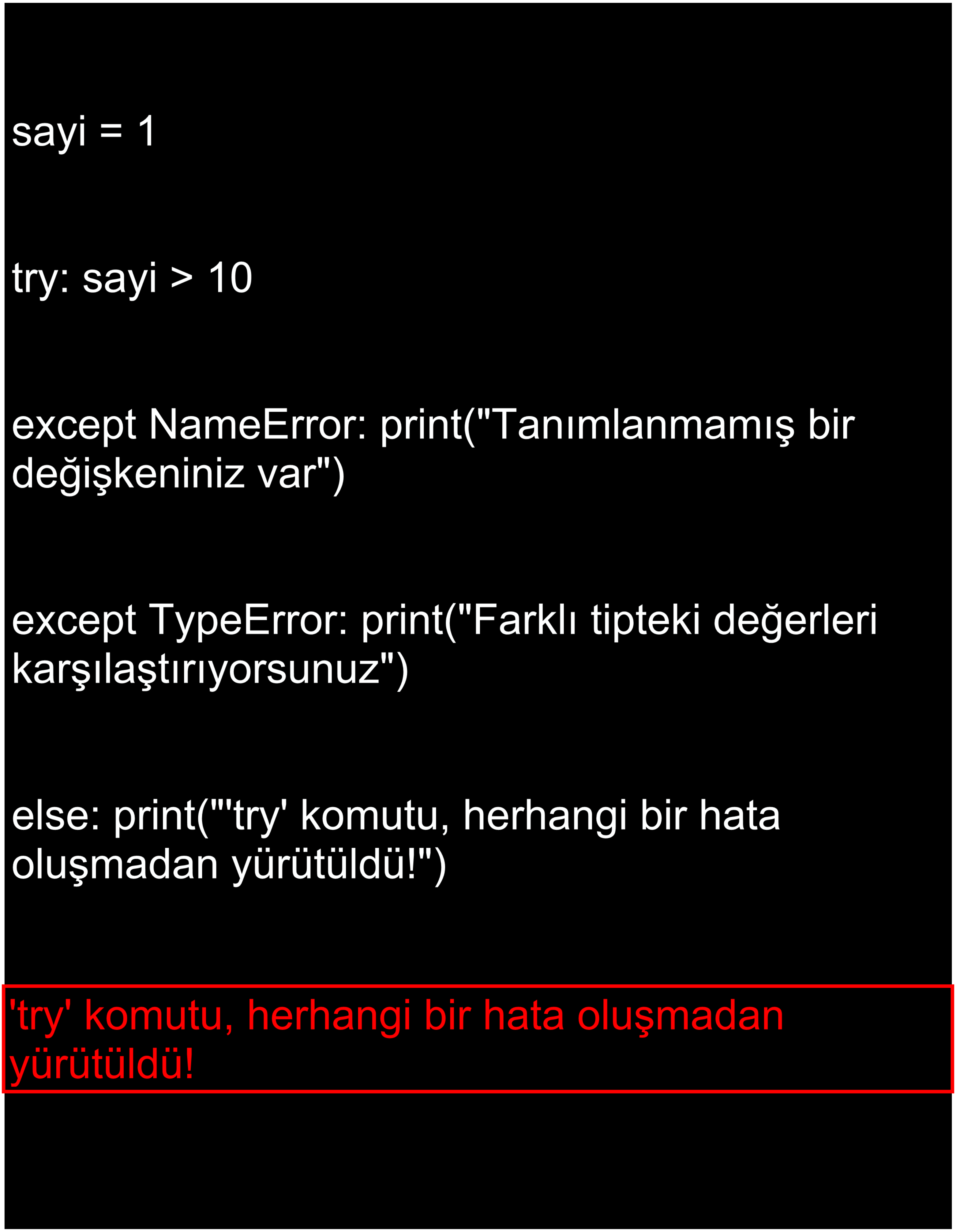
```
metin = "Örnek"

try:
        metin>10

except NameError:
        print("Tanımlanmamış bir değişkeniniz var!")

except TypeError:
        print("Farklı türdeki değerleri karşılaştırıyorsunuz!")
```

```
C:\Users\aceraspire\AppData\Local\Programs\Python\Python39\python.exe C:/Users/aceraspire/Desktop/seek_python/main.py
Farklı türdeki değerleri karşılaştırıyorsunuz!
```

try: degisken = 1/0

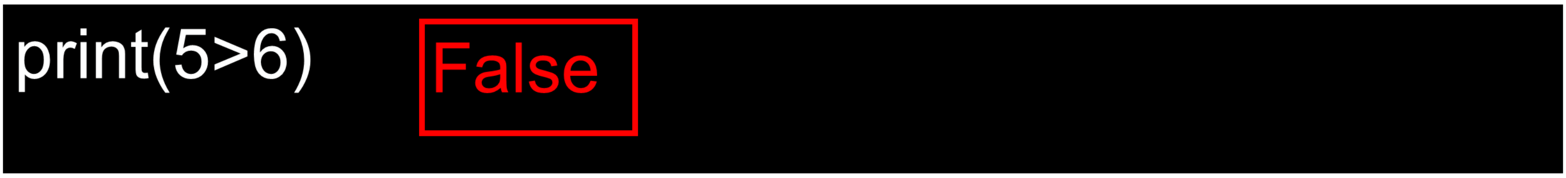
except NameError: print("Tanımlanmamış bir değişkeniniz var")
except TypeError: print("Farklı türdeki değerleri karşılaştırıyorsunuz")
except: print("Başka bir şeyler ters gitti")

Başka bir şeyler ters gitti



```
sayi = 1

try: sayi > 10

except NameError: print("Tanımlanmamış bir değişkeniniz var")

except TypeError: print("Farklı tipteki değerleri karşılaştırıyorsunuz")

else: print("'try' komutu, herhangi bir hata oluşmadan yürütüldü!")
```

```
'try' komutu, herhangi bir hata oluşmadan yürütüldü!
```



```
print(5>6)
```

```
False
```

`print(5>6)`	`False`
`print(4 in [1,2,3])`	`False`
`print("Merhaba" is "Hoşçakal")`	`False`
`print(5==6)`	`False`
`print(5==6 or 6==7)`	`False`
`print(5==6 and 6==7)`	`False`
`print("Merhaba" is not "Merhaba")`	`False`
`print(not (5==5))`	`False`
`print(3 not in [1,2,3])`	`False`

```
try:sayi>3

except: print("Birşeyler hatalı gidiyor!")

else:print("Ters giden birşey yok...")

finally:print("Bu kod bloğu, hata olsa dahi
çalışacak")
```

```
Birşeyler hatalı gidiyor!
Bu kod bloğu, hata olsa dahi çalışacak
```

```
# finally : İstisnalarla birlikte kullanılır,
# istisna olup
# olmadığına bakılmaksızın
# yürütülecek bir kod bloğudur
```

```
for dongu in range(1,9):

print(dongu)
```

```
1
2
3
4
5
6
7
8
```

```
pazarListem =
["elma","greyfurt","maydanoz","çilek"]

for listele in pazarListem:

print(listele)
```

```
elma
greyfurt
maydanoz
çilek
```

```
from  datetime import time
degiskenim=time(hour=5)
print(degiskenim)
```

```
05:00:00
```

from : Bir modülün belirli kısımlarını içe aktarmak için kullanılır

```
def fonksiyonum():
      global degisken
      degisken="Merhaba"

fonksiyonum()
print(degisken)
```

```
Merhaba
```

Bir fonksiyonun içinde global bir değişken tanımlayıp , onu fonksiyonun dışında kullandık

```python
sayi = 5

if sayi>3:
	print("Sayı, 3'ten büyüktür")
```

```
Sayı, 3'ten büyüktür
```

```python
# if : Koşullu ifade yapmak için kullanılır
```

```python
import datetime

zamaniYazdir = datetime.datetime.now()
print(zamaniYazdir)
```

```
2021-07-09 18:35:44.643370
```

```python
# Datetime modülünü içe aktardık
ve geçerli tarih ve saati görüntüledik
```

```python
listem = ["elma","muz","kiraz"]

if "muz" in listem:
	print("Evet,muz listenizde bulunuyor")
```

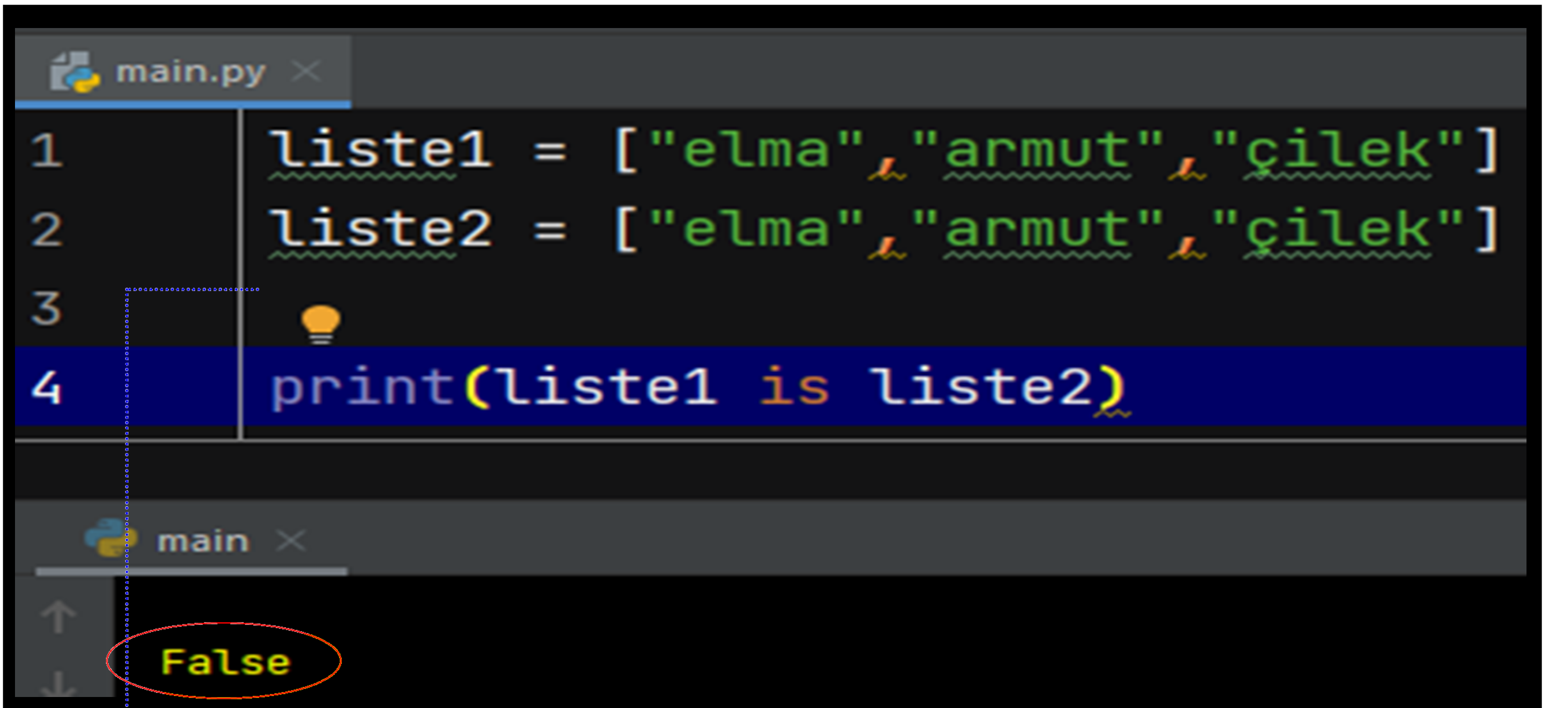
```
Evet,muz listenizde bulunuyor
```

```python
# in : Listedeki bir değerin var olup olmadığını
kontrol etmek için kullanılır
```

```python
kullaniciAdi = "User"
x=kullaniciAdi
print(x is kullaniciAdi)
```

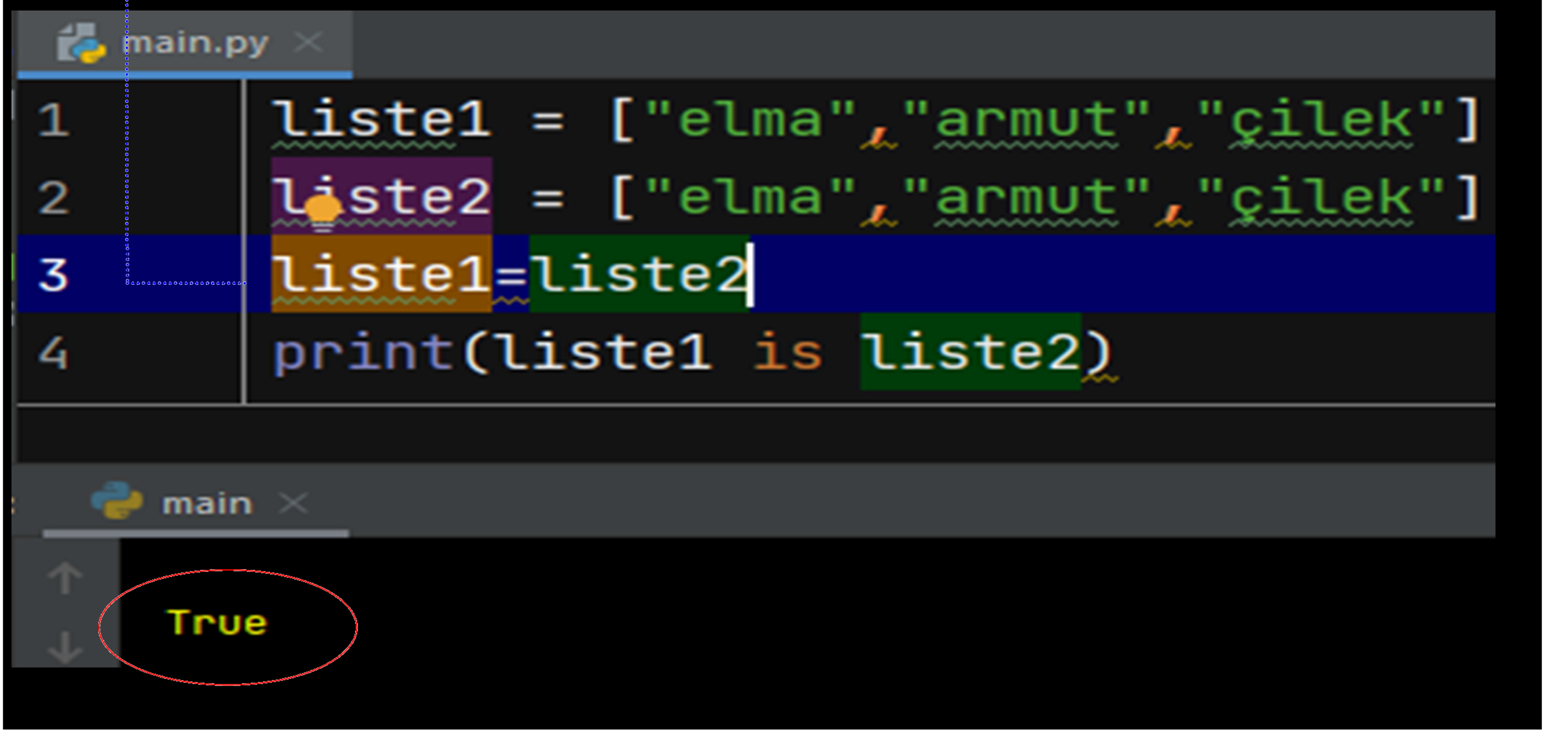
```
True
```

```python
# is : İki değişkenin eşit olup olmadığını kontrol etmek için kullanılır
```



```python
liste1 = ["elma","armut","çilek"]
liste2 = ["elma","armut","çilek"]

print(liste1 is liste2)
```

```
False
```



```python
liste1 = ["elma","armut","çilek"]
liste2 = ["elma","armut","çilek"]
liste1=liste2
print(liste1 is liste2)
```

```
True
```

```
islemYap = lambda sayi : sayi + 10
print(islemYap(5))
```

```
15
```

```
#sayi = 5
#islemYap = sayi + 10

#lambda = anahtar
#sayi = bağlı değişken

#islemYap = gövde

#lambda : bağlı değişkene, anonim işlev ataması
yapar
```

```
sayimiz = lambda x : x * 2
print(sayimiz(3))
```

```
6
```

```
sayilarimiz = lambda sayi1,sayi2,sayi3 :
sayi1+sayi2+sayi3

print(sayilarimiz(3,5,7))
```

```
15
```

```
degisken = None
print(degisken)
```

```
None
```

```
# None : Boş bir değeri temsil eder
```

```
degisken = None

if degisken is True:
      print("Değişken True değerini döndürdü")

elif degisken is False:
      print("Değiken False değerini döndürdü")

else:
      print("Değişken True veya False değeri
döndürmedi . Değişken boş.")
```

```
Değişken True veya False değeri döndürmedi .
Değişken boş.
```

```
def fonksiyon1():
    degisken = "ABC"
    def fonksiyon2():
        nonlocal degisken
        degisken = "CBA"
    fonksiyon2()
    return degisken
print(fonksiyon1())
```

```
CBA
```

```
# nonlocal : Yerel olmayan bir değişken bildirmek
için kullanılır
```

```
def fonksiyon1():
    degisken = "ABC"
    def fonksiyon2():
        degisken = "CBA"
    fonksiyon2()
    return degisken
print(fonksiyon1())
```

```
ABC
```

```
sayi = 1
print(not sayi==2)
```

```
True
```

```
sayi = 1
print(not sayi==1)
```

```
False
```

```
deger = False
print(not deger)
```

```
True
```

```
sayi = (5>3 or 5>10)
print(sayi)
```

```
True
```

```
metnimiz = "Ornek"
harfSay=len(metnimiz)

if(harfSay<=5 or harfSay<=10):
    print(metnimiz+" kelimesi 5-10 karakter aralığında")

elif(harfSay<5 or harfSay>5):
    print(metnimiz+" kelimesi 5 karakterden az veya 5 karakterden fazla")

else:
    print("Hatalı durum oluştu...")
```

```
Ornek kelimesi 5-10 karakter aralığında
```

```
if 10 > 11 or 10 > 9:
    print("İfadelerden en az 1 tanesi doğru")

else:
    print("İfadelerden hiçbiri doğru değil")
```

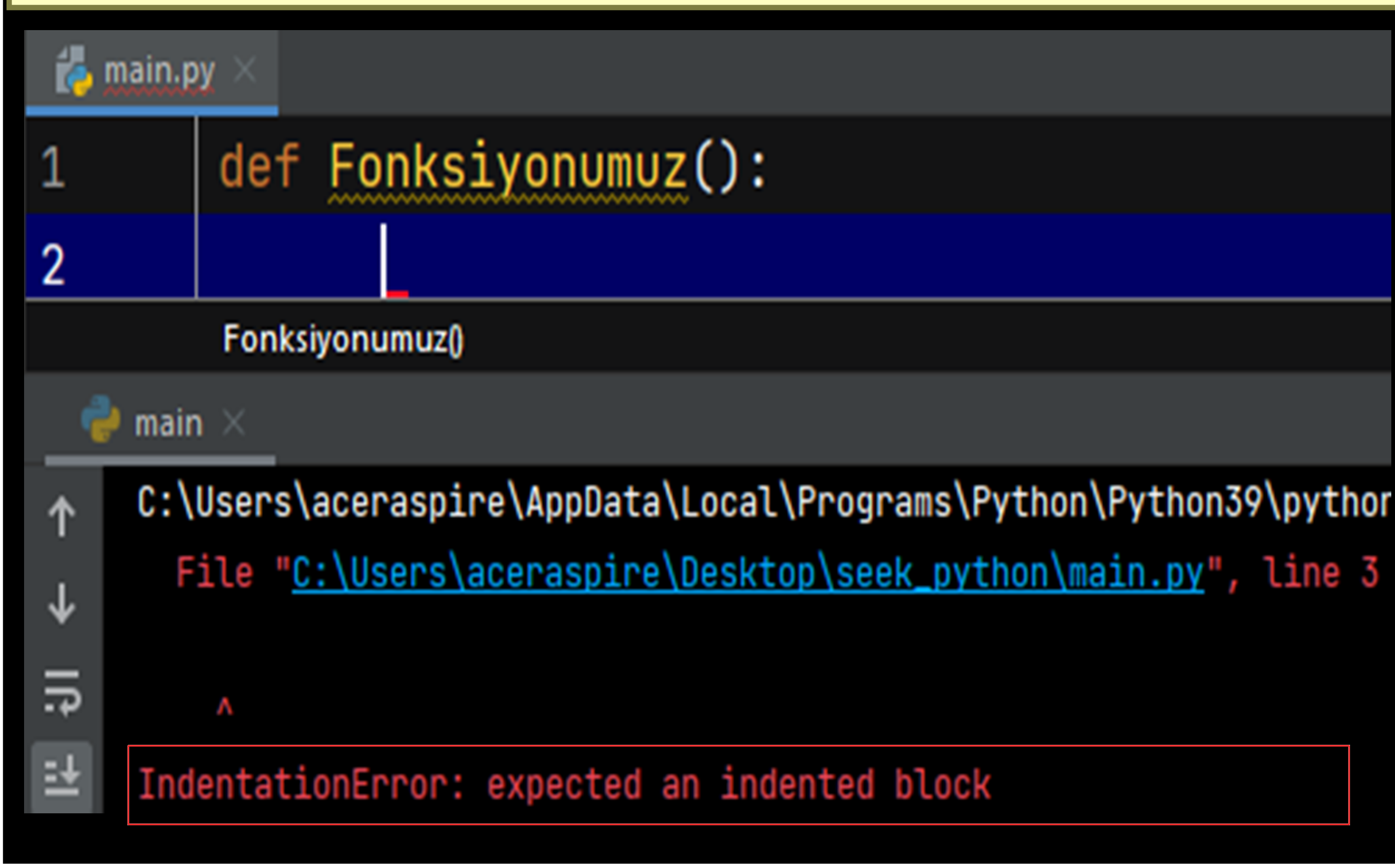
```
İfadelerden en az 1 tanesi doğru
```

```
for sayilar in [0,1,2]:
      pass
```

```
# pass : Boş bir ifade & hiçbir şey yapmayacak bir ifade

# Gelecekteki kod için bir yer tutucu oluşturur

# fonksiyon tanımlarında, sınıf tanımlarında veya if deyimlerinde boş koda izin verilmez !
```

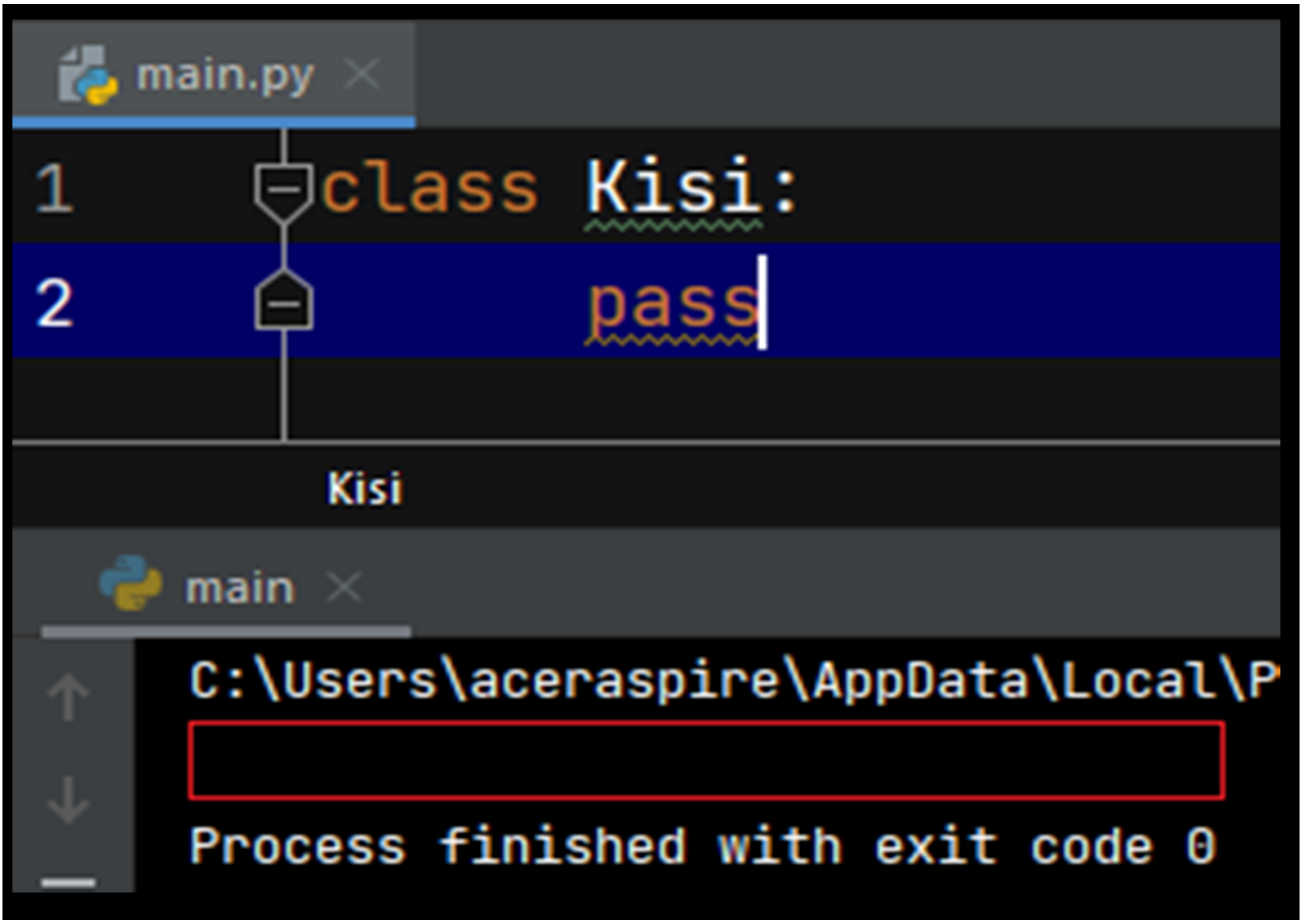
```
def fonksiyonumuz():
      pass
```

```
# Pass ifadesi olmasaydı, bir hata mesajı döndürürdü
```



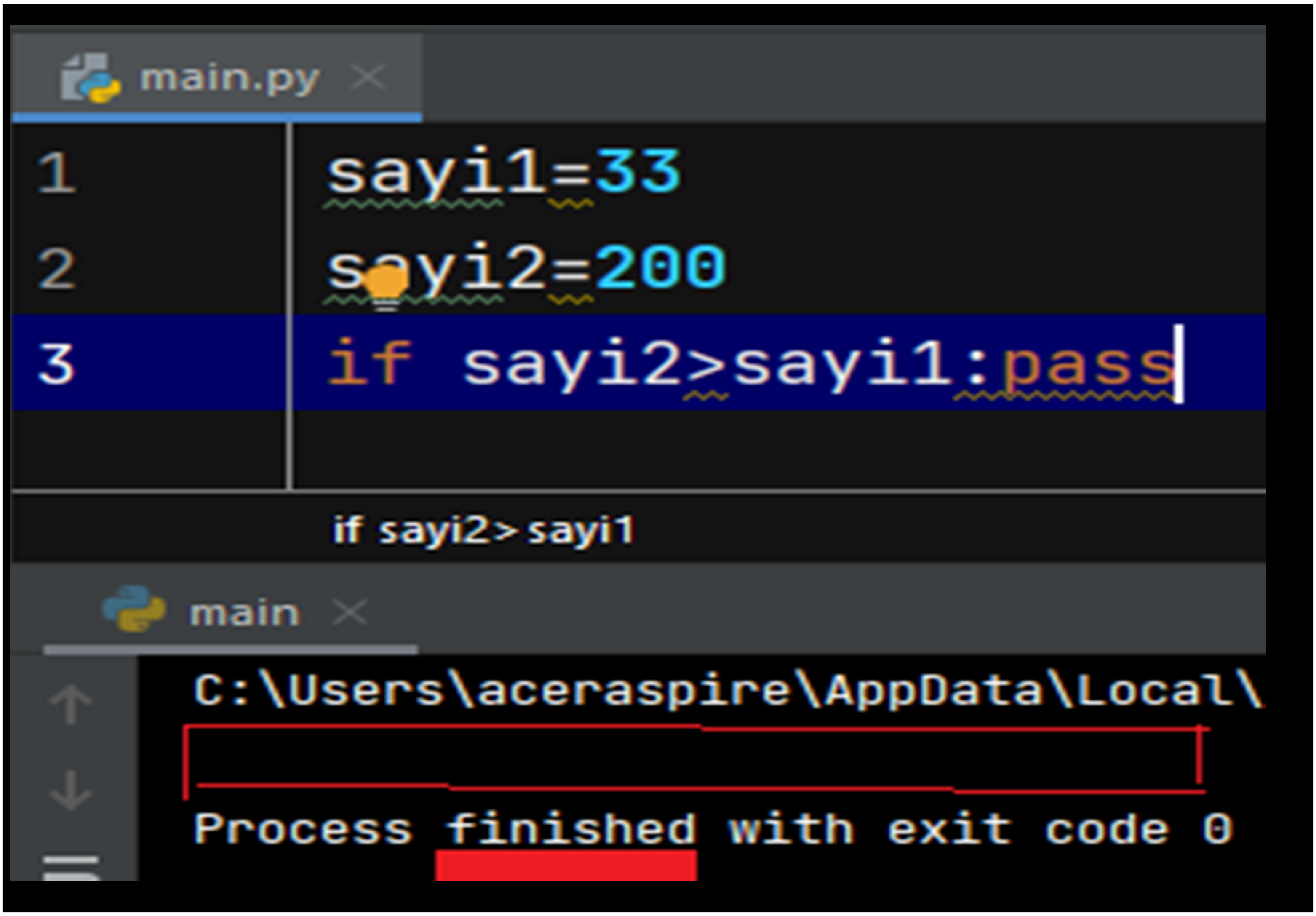
```
def Fonksiyonumuz():
    
```

```
C:\Users\aceraspire\AppData\Local\Programs\Python\Python39\pythor
  File "C:\Users\aceraspire\Desktop\seek_python\main.py", line 3


    ^
IndentationError: expected an indented block
```


main.py
Kisi
main

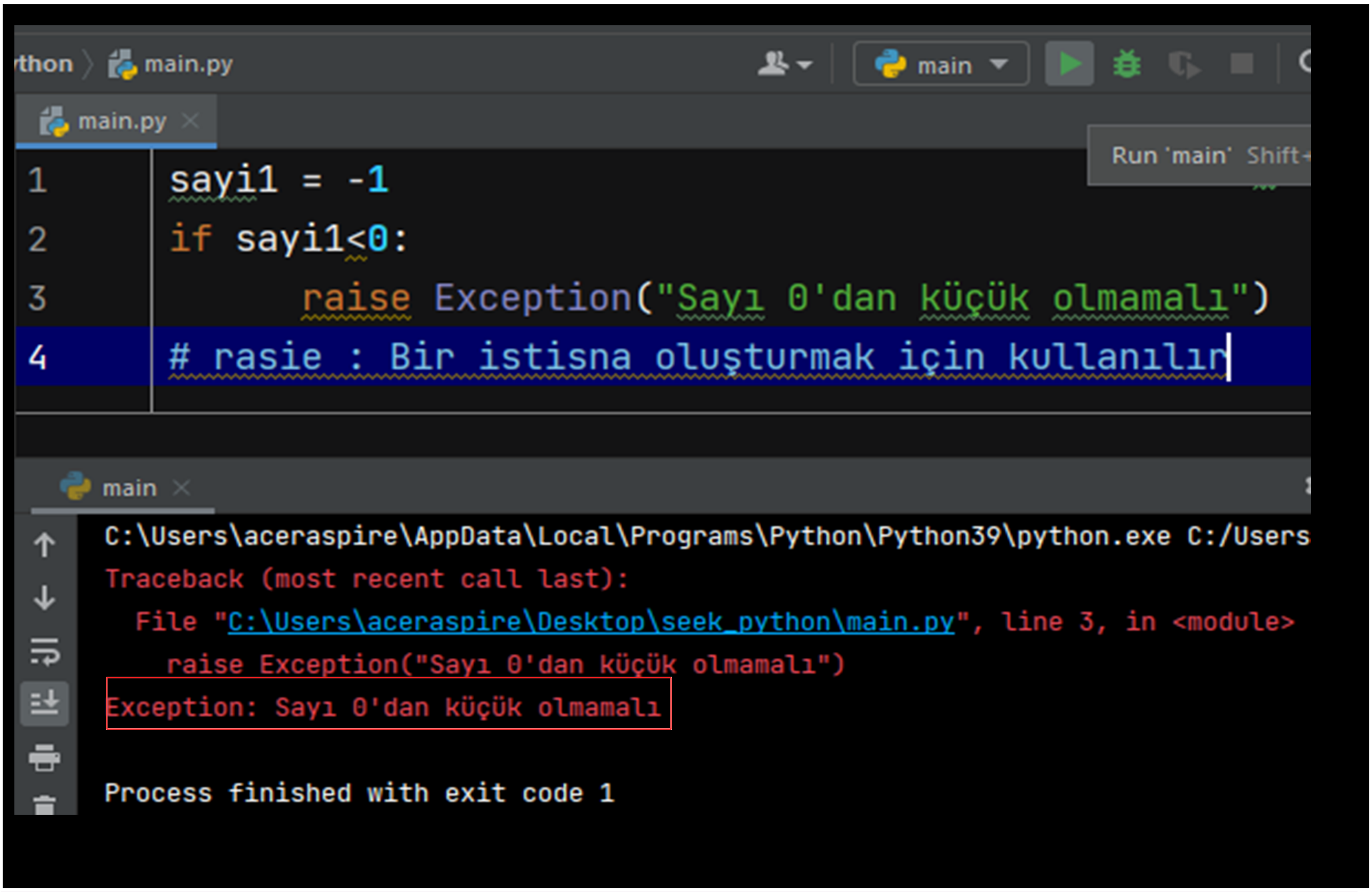
```python
class Kisi:
        pass
```

```
C:\Users\aceraspire\AppData\Local\P

Process finished with exit code 0
```


main.py
if sayi2>sayi1
main

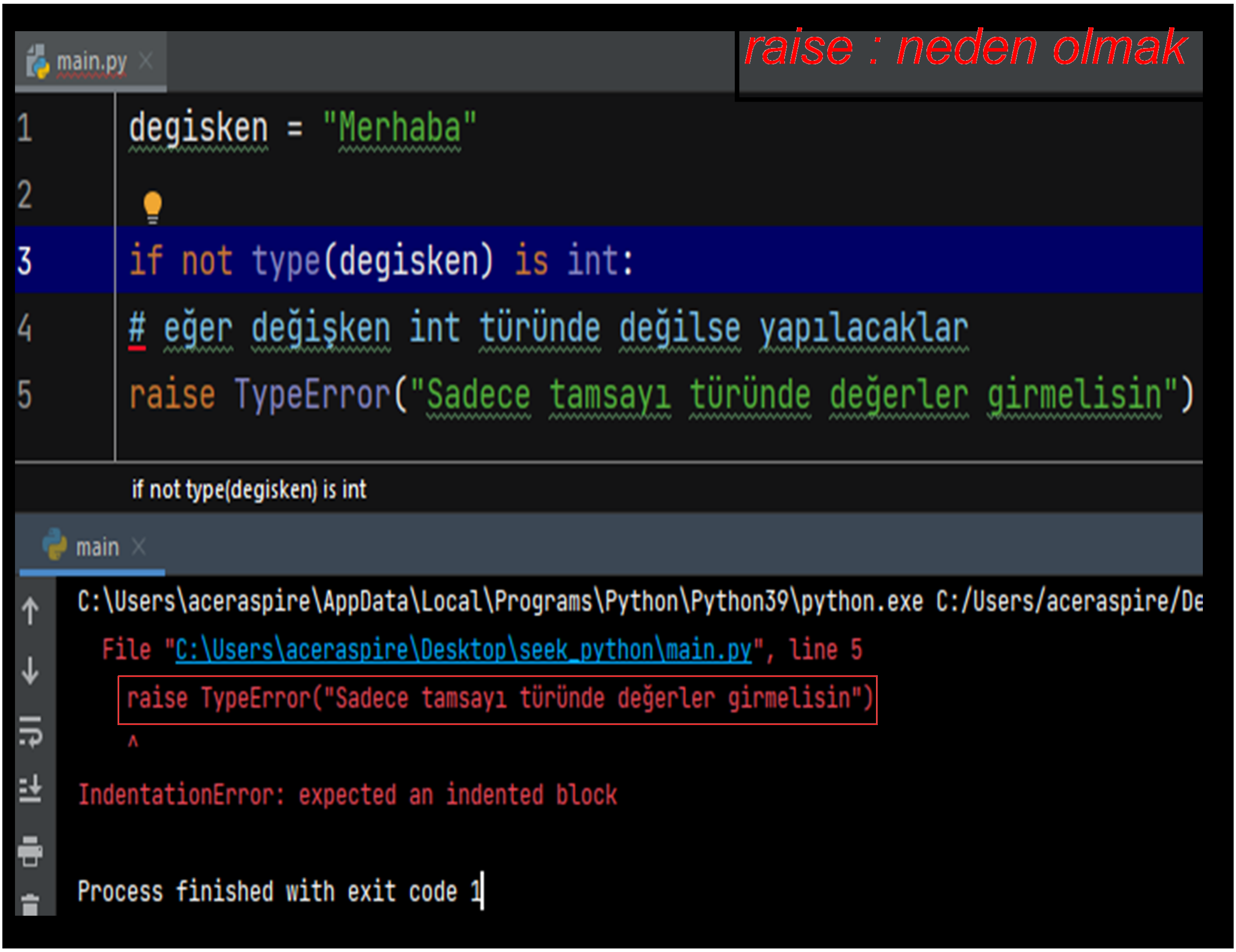
```python
sayi1=33
sayi2=200
if sayi2>sayi1:pass
```

```
C:\Users\aceraspire\AppData\Local\

Process finished with exit code 0
```



```python
sayi1 = -1
if sayi1<0:
      raise Exception("Sayı 0'dan küçük olmamalı")
# rasie : Bir istisna oluşturmak için kullanılır
```

```
C:\Users\aceraspire\AppData\Local\Programs\Python\Python39\python.exe C:/Users
Traceback (most recent call last):
  File "C:\Users\aceraspire\Desktop\seek_python\main.py", line 3, in <module>
    raise Exception("Sayı 0'dan küçük olmamalı")
Exception: Sayı 0'dan küçük olmamalı

Process finished with exit code 1
```



raise : neden olmak

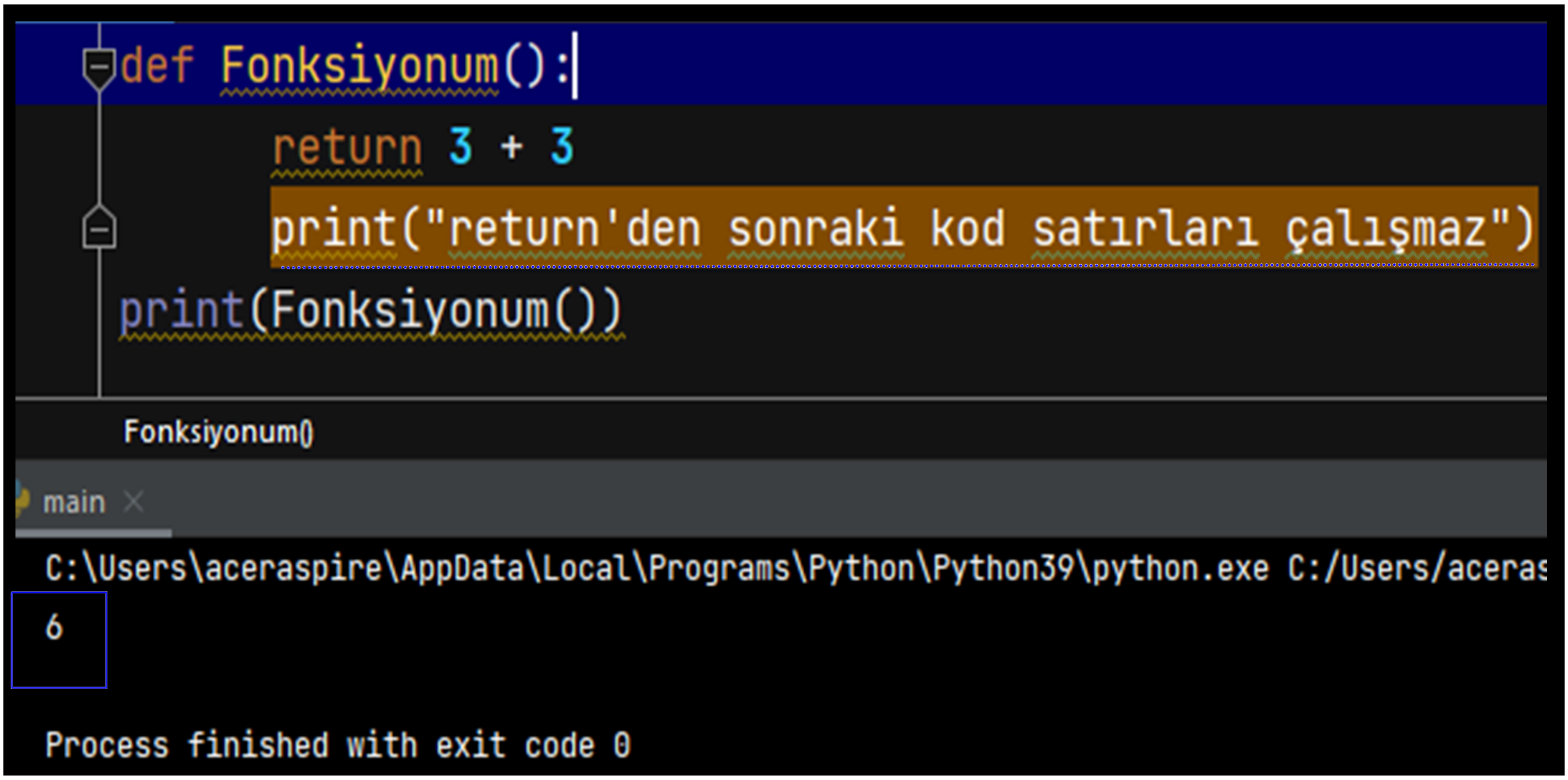
```python
degisken = "Merhaba"

if not type(degisken) is int:
# eğer değişken int türünde değilse yapılacaklar
raise TypeError("Sadece tamsayı türünde değerler girmelisin")
```

```
C:\Users\aceraspire\AppData\Local\Programs\Python\Python39\python.exe C:/Users/aceraspire/De
  File "C:\Users\aceraspire\Desktop\seek_python\main.py", line 5
    raise TypeError("Sadece tamsayı türünde değerler girmelisin")
    ^
IndentationError: expected an indented block

Process finished with exit code 1
```

```
def islemYap():
    return 3+3
print(islemYap())
```

```
6
```

return : Bir fonksiyondan çıkmak ve bir değer döndürmek için kullanılır



```
def Fonksiyonum():
        return 3 + 3
        print("return'den sonraki kod satırları çalışmaz")
print(Fonksiyonum())
```

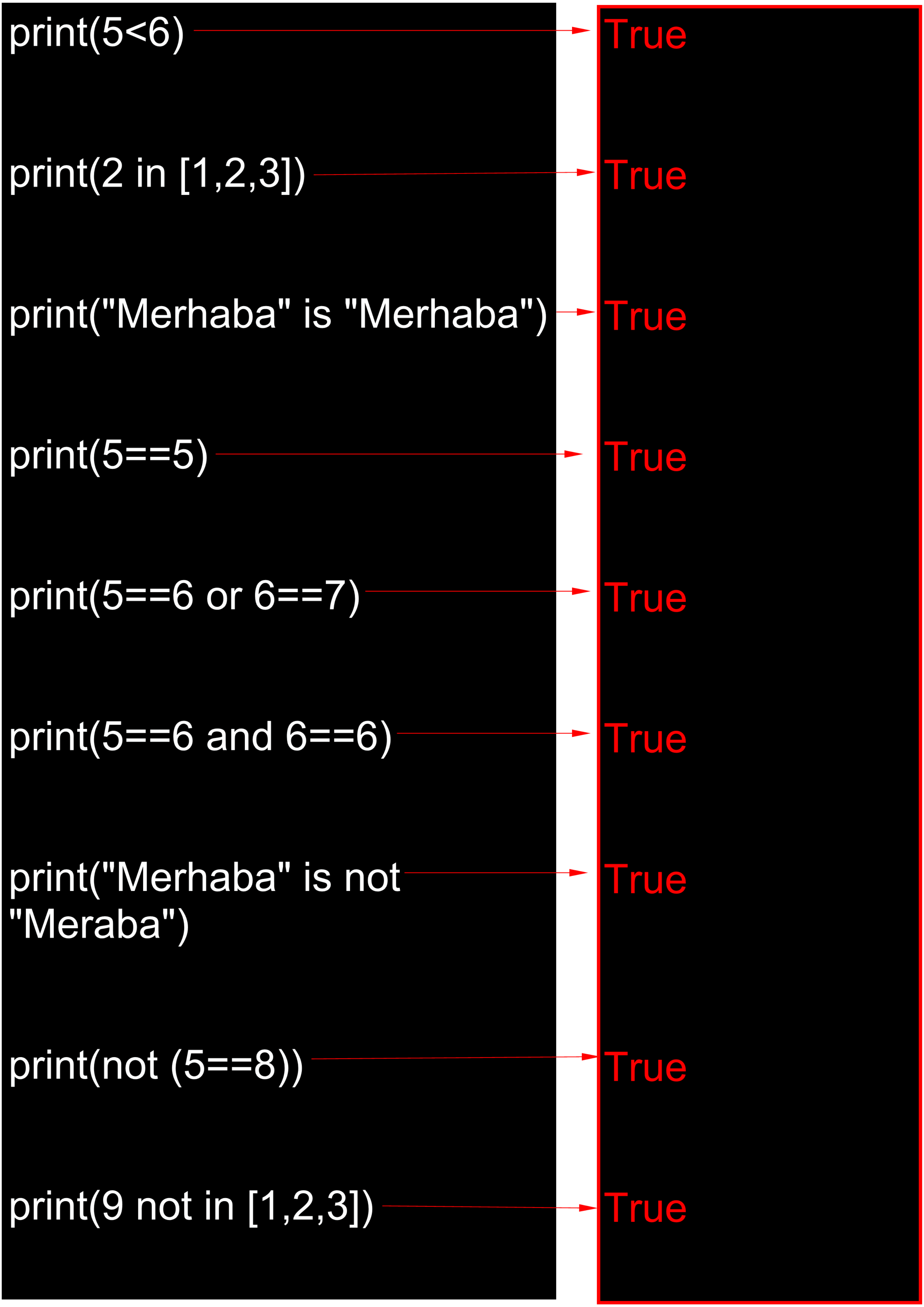
```
C:\Users\aceraspire\AppData\Local\Programs\Python\Python39\python.exe C:/Users/aceras
6

Process finished with exit code 0
```

```
print(7>6)
```

```
True
```

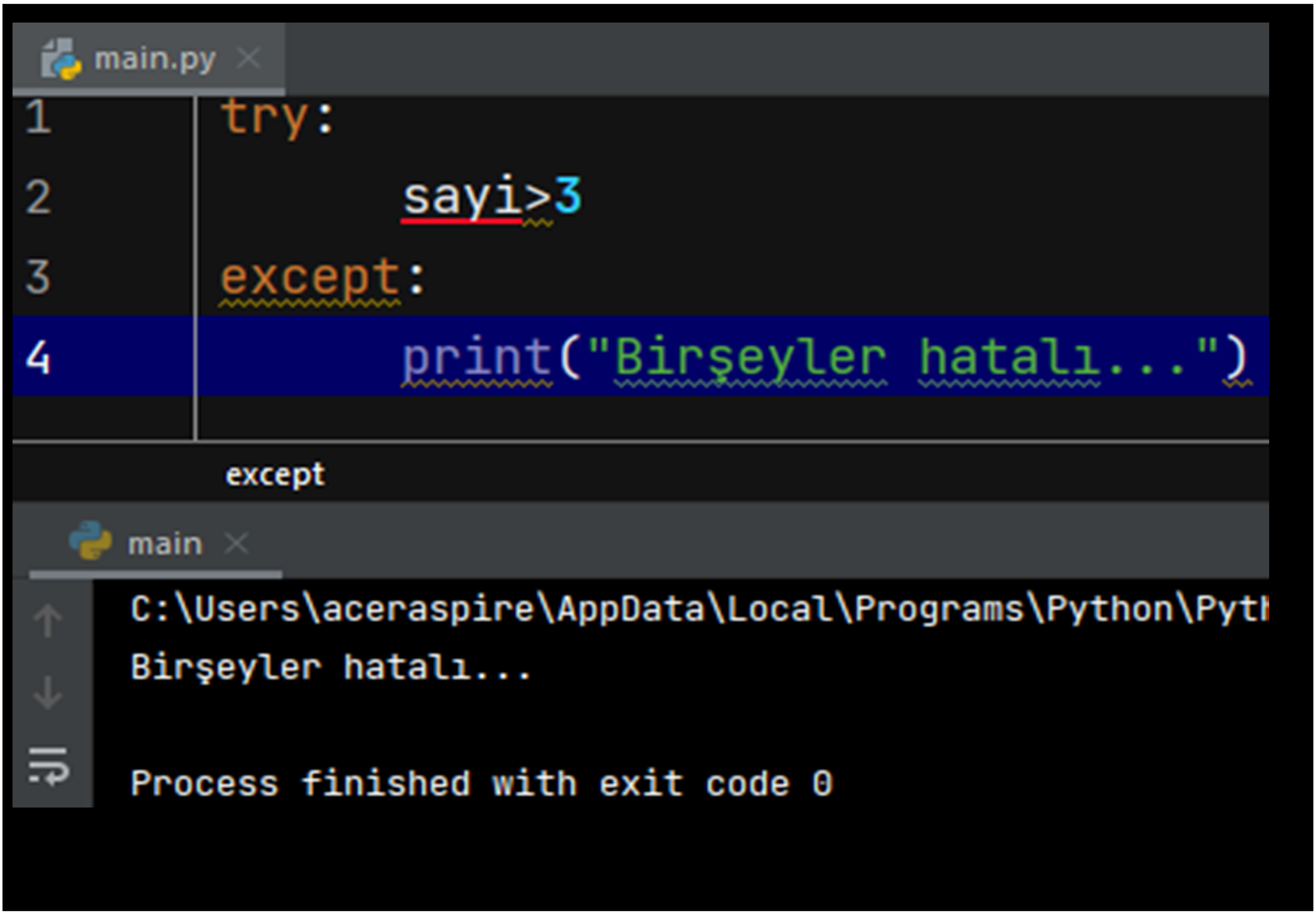
boolean : Karşılaştırma işlemlerinde kullanılır

boolean : True ya da False değeri döndürür



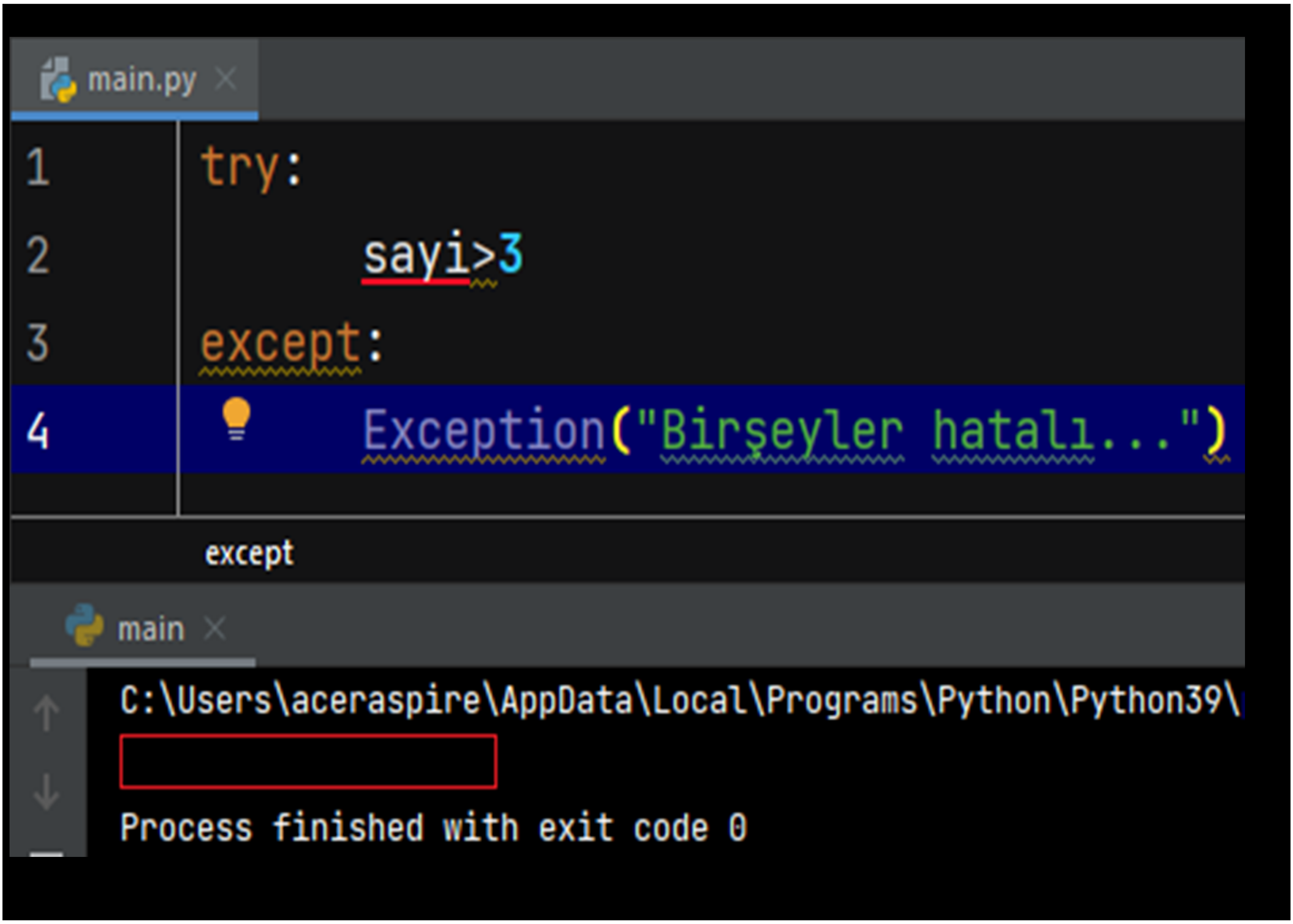
```
print(5<6)

print(2 in [1,2,3])

print("Merhaba" is "Merhaba")

print(5==5)

print(5==6 or 6==7)

print(5==6 and 6==6)

print("Merhaba" is not
"Meraba")

print(not (5==8))

print(9 not in [1,2,3])
```

```
True

True

True

True

True

True

True

True

True
```


main.py
except
main

```python
try:
        sayi>3
except:
        print("Birşeyler hatalı...")
```

```
C:\Users\aceraspire\AppData\Local\Programs\Python\Pyt
Birşeyler hatalı...

Process finished with exit code 0
```

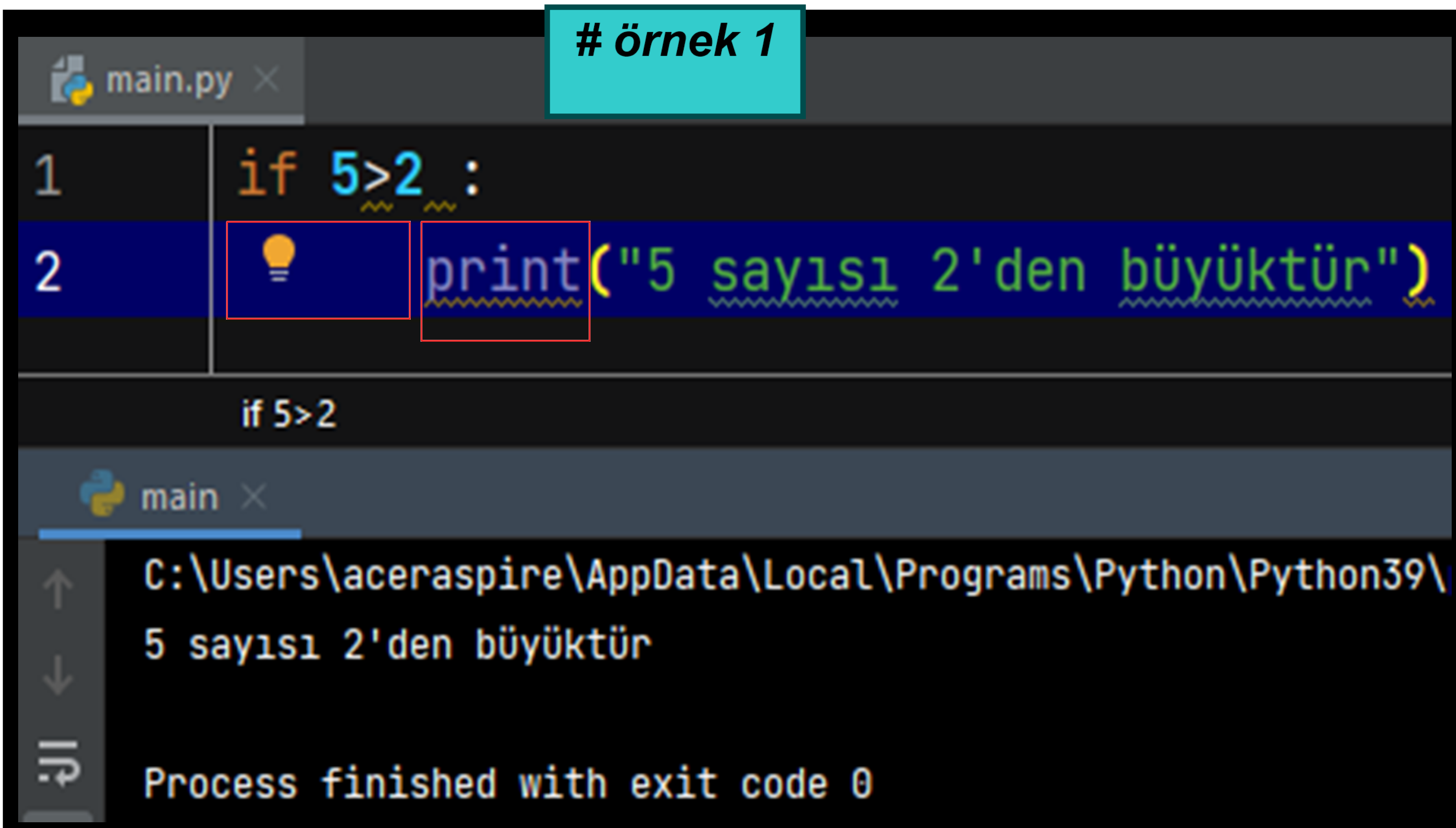

main.py
except
main

```python
try:
        sayi>3
except:
        Exception("Birşeyler hatalı...")
```

```
C:\Users\aceraspire\AppData\Local\Programs\Python\Python39\

Process finished with exit code 0
```

```
sayi = 0

while sayi<9:      #bu koşul sağlanana kadar aşağıdaki kodları çalıştır

    print(sayi)    #0 değerini yazdır ve

    sayi=sayi+1    #şimdi de 0 değerini 1 artır, sonucu da sayı değişkenine eşitle ve koşul sağlandı mı kontrol et, sağlanmadıysa başa dön
```

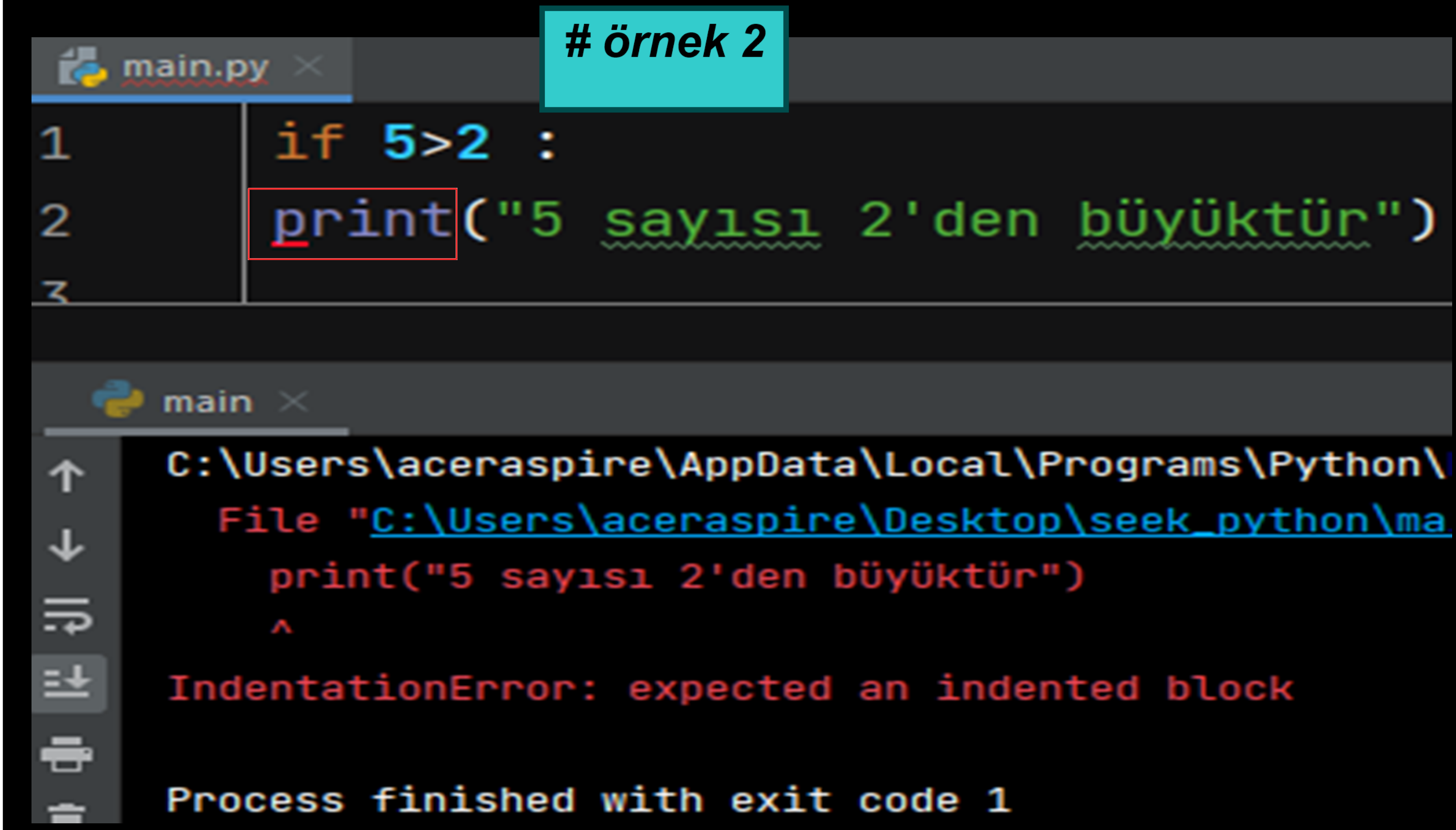
```
0
1
2
3
4
5
6
7
8
```



örnek 1

```
if 5>2 :
    print("5 sayısı 2'den büyüktür")
```

```
C:\Users\aceraspire\AppData\Local\Programs\Python\Python39\
5 sayısı 2'den büyüktür

Process finished with exit code 0
```



örnek 2

```
if 5>2 :
print("5 sayısı 2'den büyüktür")
```

```
C:\Users\aceraspire\AppData\Local\Programs\Python\
  File "C:\Users\aceraspire\Desktop\seek_python\ma
    print("5 sayısı 2'den büyüktür")
    ^
IndentationError: expected an indented block

Process finished with exit code 1
```

Indentation (Girinti) : Bir kod satırının başındaki boşlukları ifade eder.

```
# Yorum satırları, çalışmayacak kod satırlarıdır

# Bu, tek yorum satırıdır , diyez işareti satır başına
koyulur, altgr+3 kombinasyonuyla oluşturulur

""" Bu çoklu
yorum
satırıdır , 3 çift tırnak arasına yazılan çoklu satır
yorumudur
""""
```

```
degiskenim="Merhaba"
degis_kenim="Merhaba"
_degis_kenim="Merhaba"
degisKenim="Merhaba"
DEGISKENIM="Merhaba"
degiskenim2="Merhaba"
```

→ #Mümkün olan değişken atamaları

```
2degiskenim="Merhaba"
degis-kenim="Merhaba"
degisken im="Merhaba"
```

→ #Mümkün olmayan değişken atamaları

```
degisken1, degisken2, degisken3 = "1","2","3"

print(degisken1)
print(degisken2)
print(degisken3)
```

```
1
2
3
```

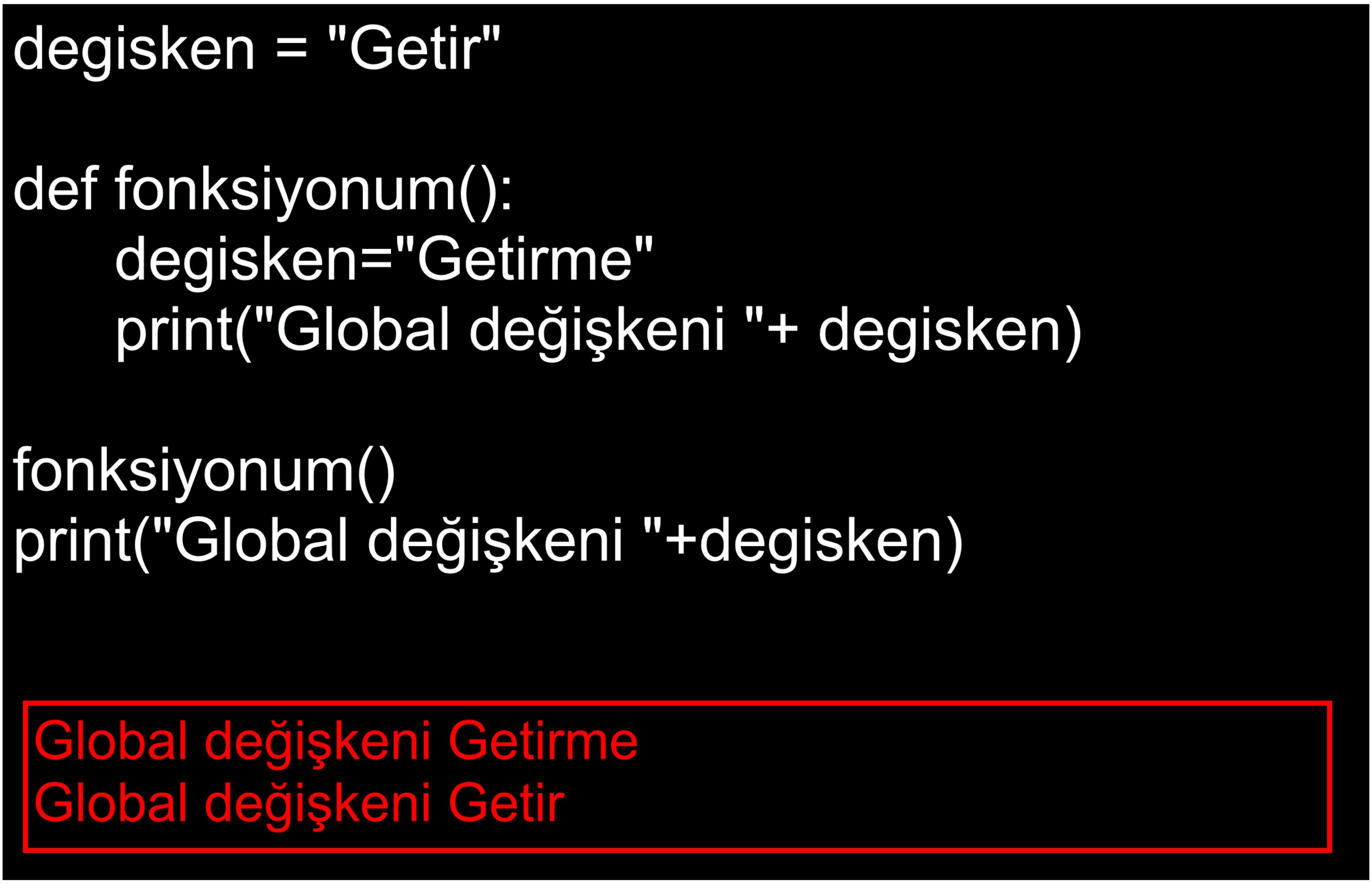
```
# Çoklu değişken ataması bu şekilde yapılabilir
```

```
degisken1=degisken2=degisken3 = "A"

print(degisken1)
print(degisken2)
print(degisken3)
```

```
A
A
A
```

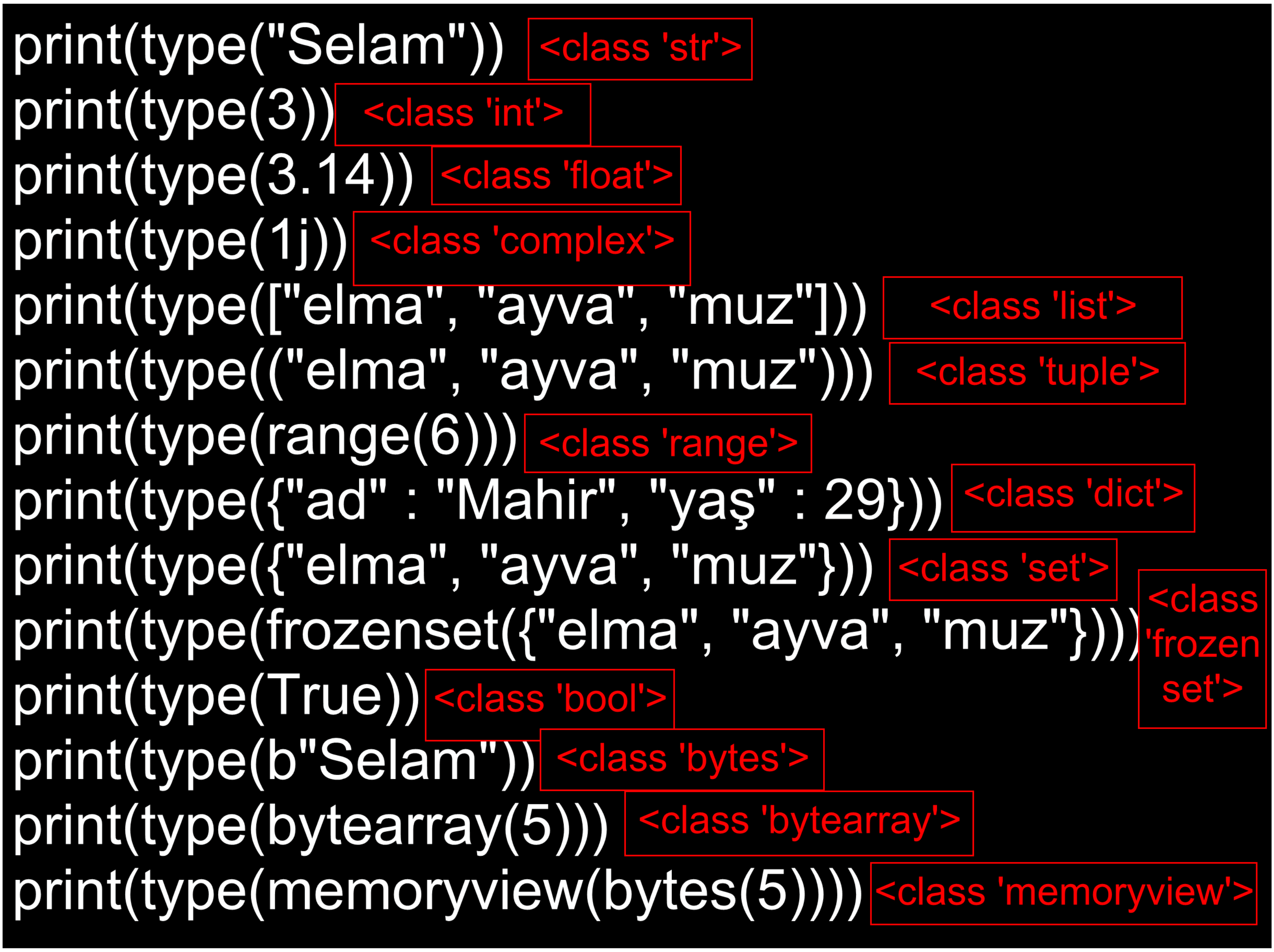
```
degisken = "Getir"

def fonksiyonum():
    print("Global değişkeni "+ degisken)

fonksiyonum()
```

```
Global değişkeni Getir
```



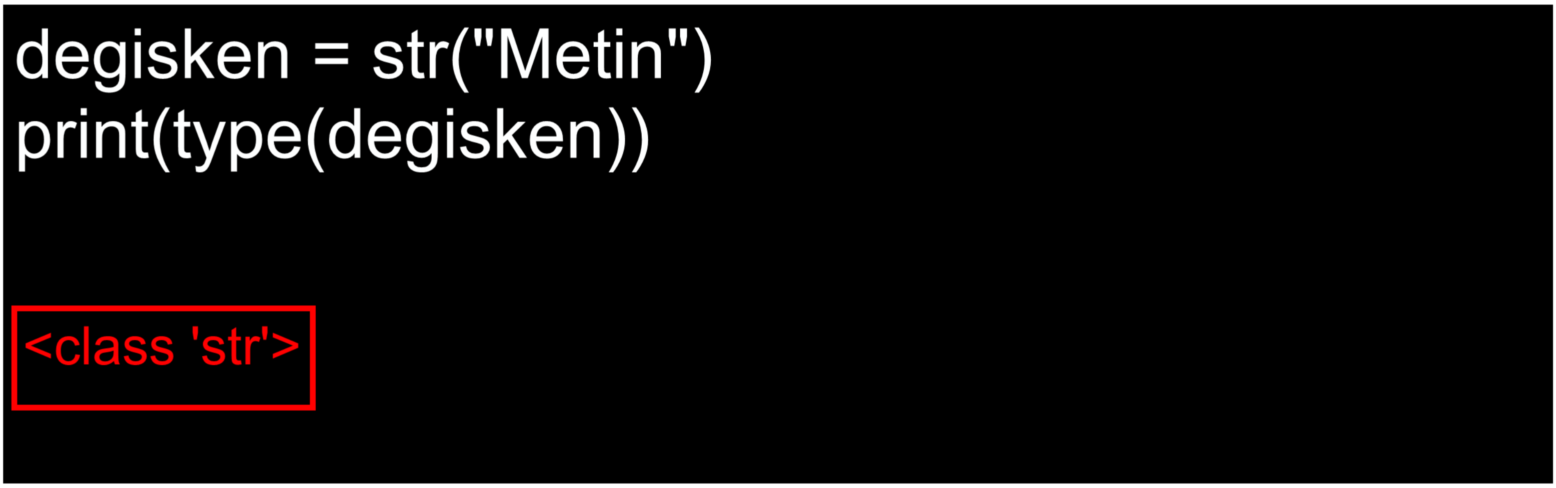
```
degisken = "Getir"

def fonksiyonum():
    degisken="Getirme"
    print("Global değişkeni "+ degisken)

fonksiyonum()
print("Global değişkeni "+degisken)
```

```
Global değişkeni Getirme
Global değişkeni Getir
```



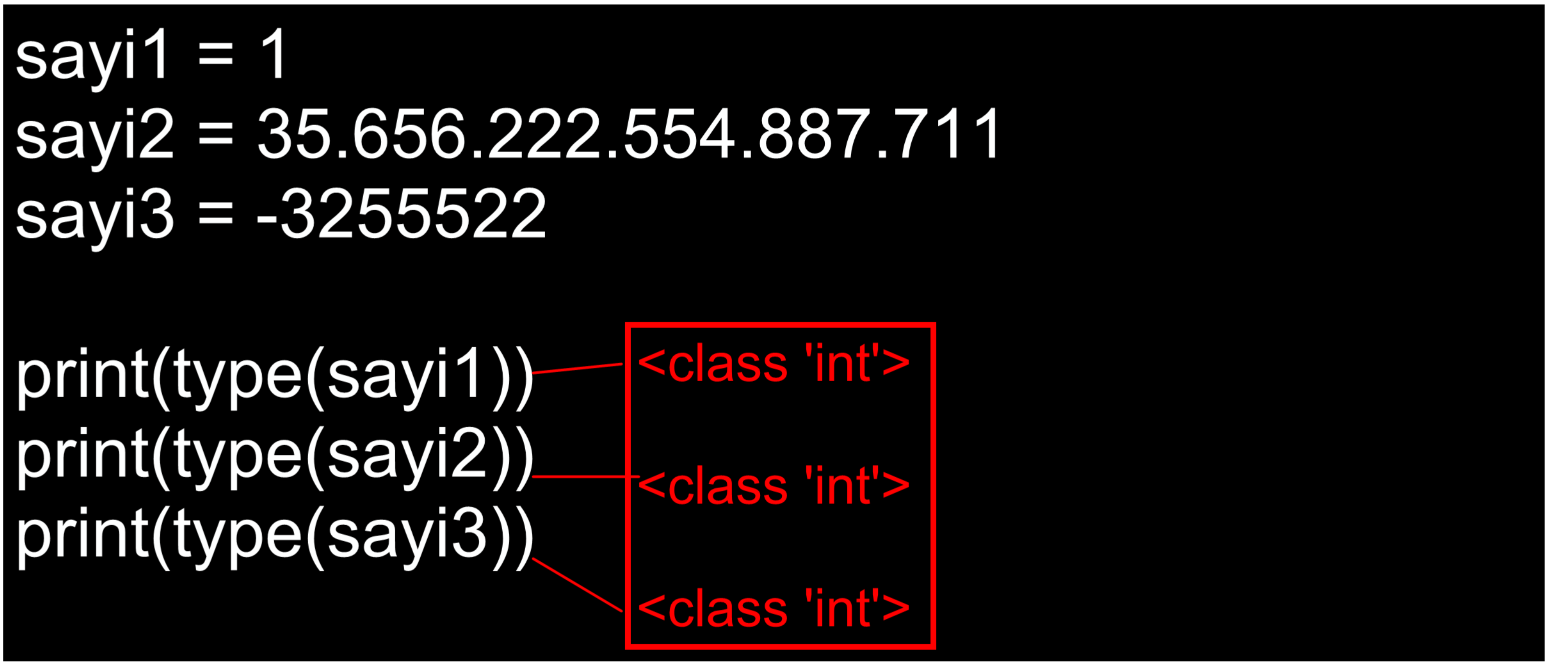
```
print(type("Selam"))
print(type(3))
print(type(3.14))
print(type(1j))
print(type(["elma", "ayva", "muz"]))
print(type(("elma", "ayva", "muz")))
print(type(range(6)))
print(type({"ad" : "Mahir", "yaş" : 29}))
print(type({"elma", "ayva", "muz"}))
print(type(frozenset({"elma", "ayva", "muz"})))
print(type(True))
print(type(b"Selam"))
print(type(bytearray(5)))
print(type(memoryview(bytes(5))))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
<class 'complex'>
<class 'list'>
<class 'tuple'>
<class 'range'>
<class 'dict'>
<class 'set'>
<class 'frozenset'>
<class 'bool'>
<class 'bytes'>
<class 'bytearray'>
<class 'memoryview'>
```



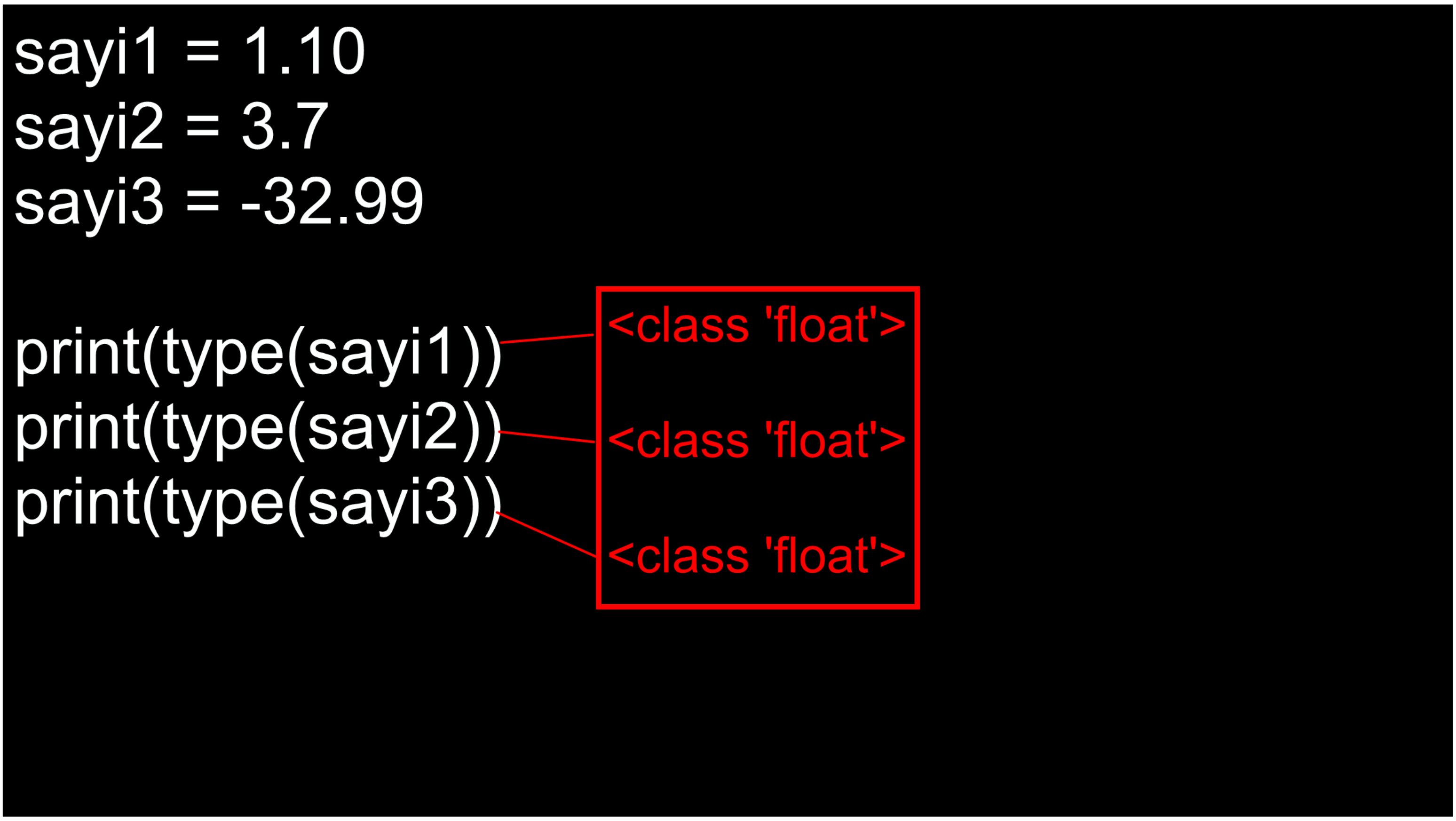
```
degisken = str("Metin")
print(type(degisken))
```

```
<class 'str'>
```



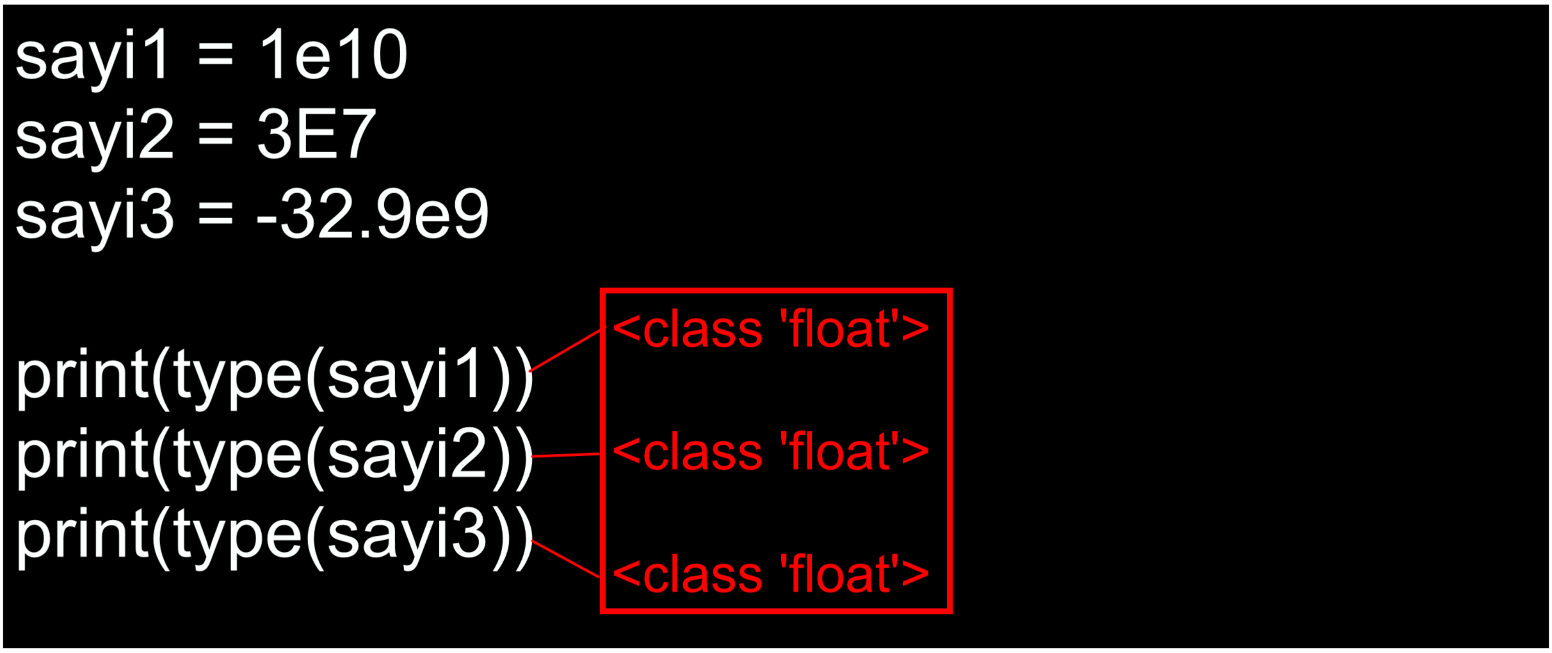
```
sayi1 = 1
sayi2 = 35.656.222.554.887.711
sayi3 = -3255522

print(type(sayi1))
print(type(sayi2))
print(type(sayi3))
```

```
<class 'int'>

<class 'int'>

<class 'int'>
```



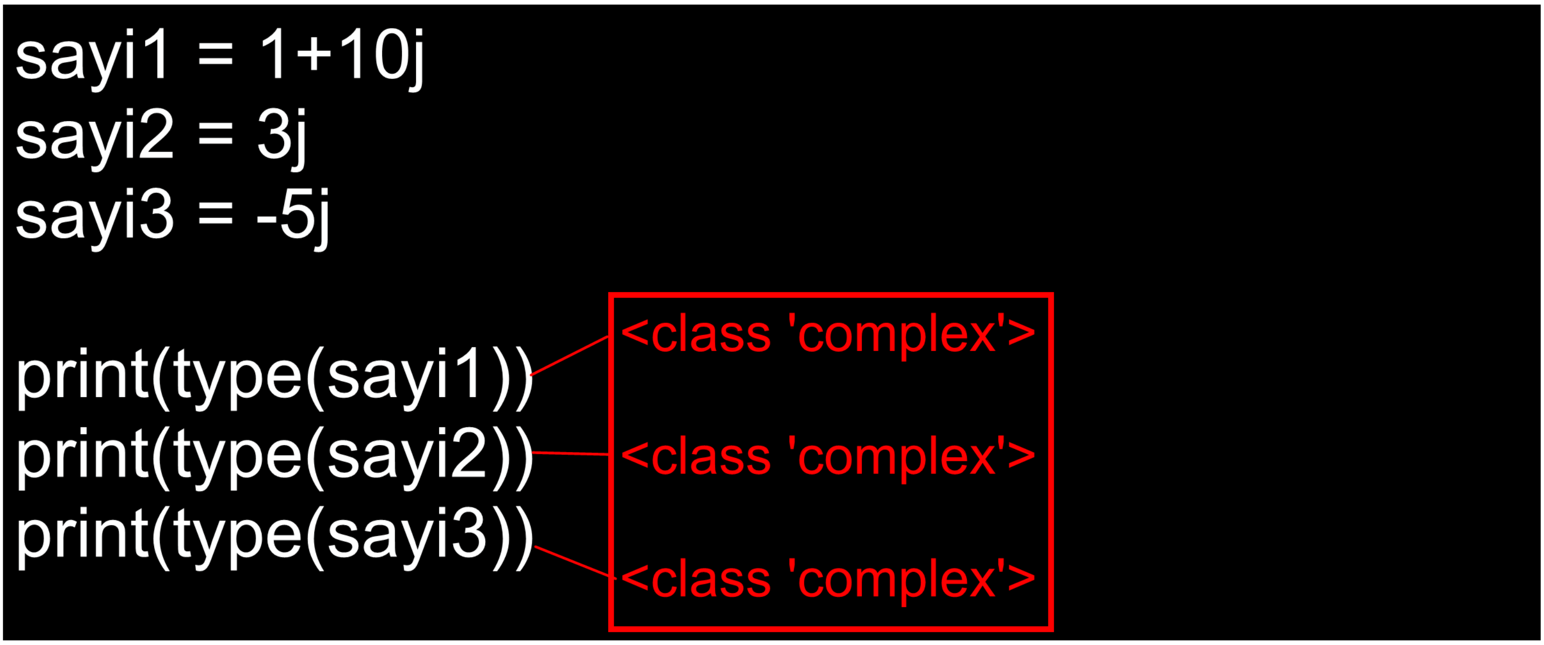
```
sayi1 = 1.10
sayi2 = 3.7
sayi3 = -32.99

print(type(sayi1))
print(type(sayi2))
print(type(sayi3))
```

```
<class 'float'>

<class 'float'>

<class 'float'>
```



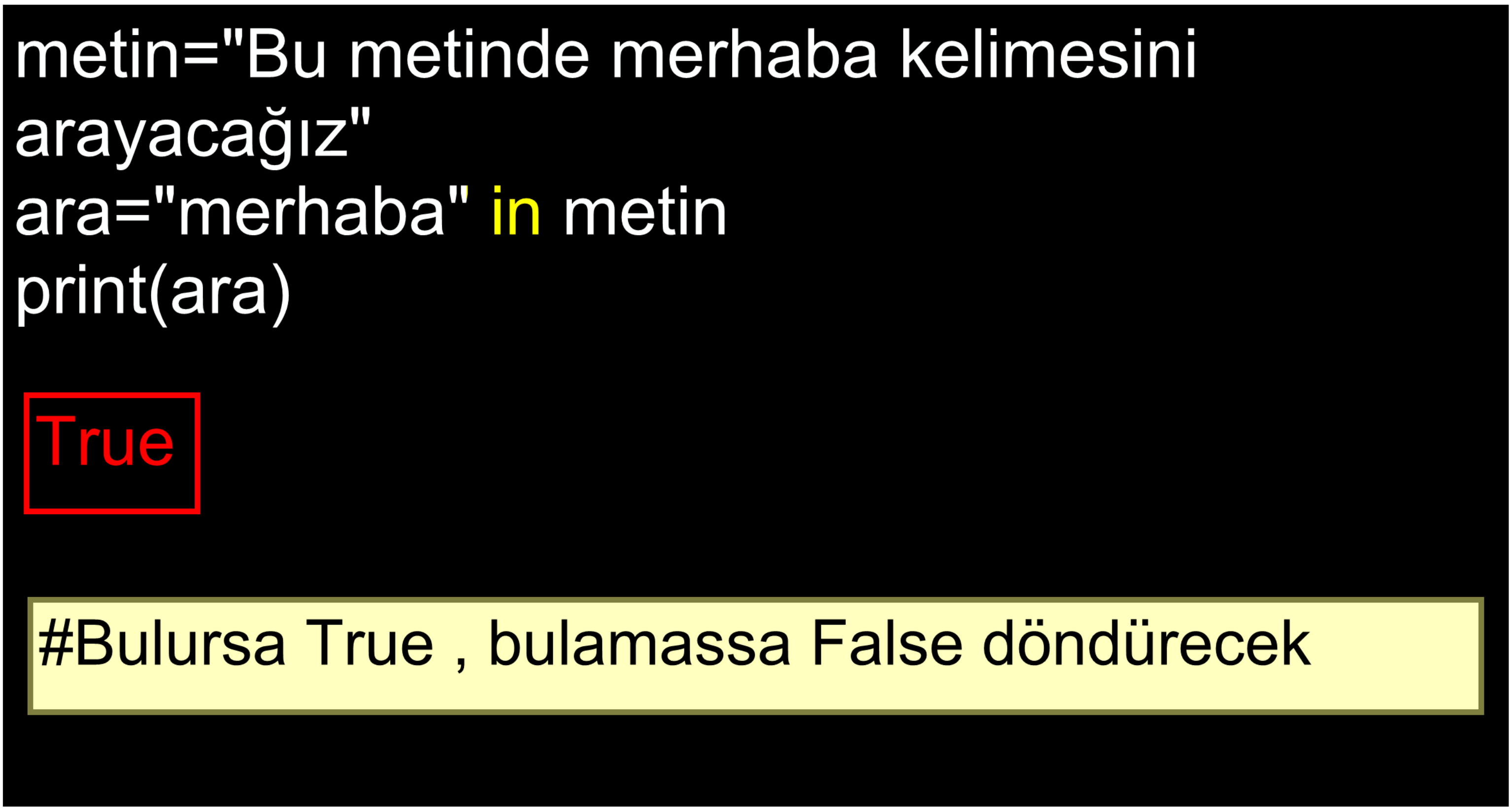
```
sayi1 = 1e10
sayi2 = 3E7
sayi3 = -32.9e9

print(type(sayi1))
print(type(sayi2))
print(type(sayi3))
```

```
<class 'float'>
<class 'float'>
<class 'float'>
```



```
sayi1 = 1+10j
sayi2 = 3j
sayi3 = -5j

print(type(sayi1))
print(type(sayi2))
print(type(sayi3))
```

```
<class 'complex'>
<class 'complex'>
<class 'complex'>
```



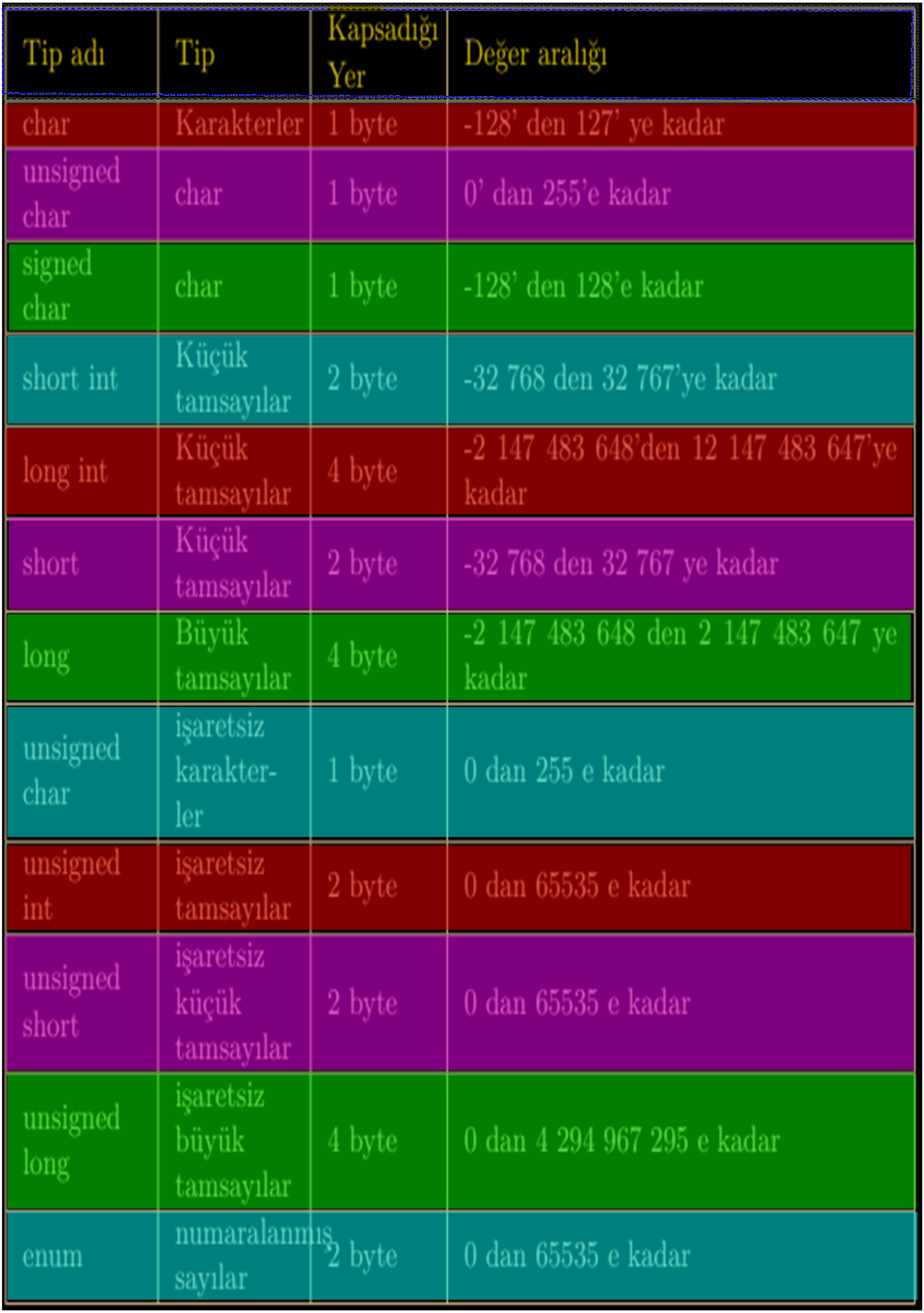
```
metin="Bu metinde merhaba kelimesini
arayacağız"
ara="merhaba" in metin
print(ara)
```

```
True
```

```
#Bulursa True , bulamassa False döndürecek
```

```
tamSayi=1
virgulluSayi=2.9
komplexSayi=5j

tamSayiDonustur_Floata=float(tamSayi)
virgulluSayiyiDonusturTamSayiya=int(virgulluSayi)
tamSayiDonustur_Komplexe=complex(tamSayi)

print(tamSayiDonustur_Floata)
print(virgulluSayiyiDonusturTamSayiya)
print(tamSayiDonustur_Komplexe)

print(type(tamSayiDonustur_Floata))
print(type(virgulluSayiyiDonusturTamSayiya))
print(type(tamSayiDonustur_Komplexe))
```

```
1.0
2
(1+0j)
<class 'float'>
<class 'int'>
<class 'complex'>
```



Tip adı	Tip	Kapsadığı Yer	Değer aralığı
char	Karakterler	1 byte	-128' den 127' ye kadar
unsigned char	char	1 byte	0' dan 255'e kadar
signed char	char	1 byte	-128' den 128'e kadar
short int	Küçük tamsayılar	2 byte	-32 768 den 32 767'ye kadar
long int	Küçük tamsayılar	4 byte	-2 147 483 648'den 12 147 483 647'ye kadar
short	Küçük tamsayılar	2 byte	-32 768 den 32 767 ye kadar
long	Büyük tamsayılar	4 byte	-2 147 483 648 den 2 147 483 647 ye kadar
unsigned char	işaretsiz karakterler	1 byte	0 dan 255 e kadar
unsigned int	işaretsiz tamsayılar	2 byte	0 dan 65535 e kadar
unsigned short	işaretsiz küçük tamsayılar	2 byte	0 dan 65535 e kadar
unsigned long	işaretsiz büyük tamsayılar	4 byte	0 dan 4 294 967 295 e kadar
enum	numaralanmış sayılar	2 byte	0 dan 65535 e kadar

```
metin="Bu metinde merhaba kelimesini
arayacağız"
ara="merhaba" not in metin
print(ara)
```

```
False
```

```
#Bulursa False , bulamassa True döndürecek
```

```
print(bool("Metin"))

print(bool(20))

print(bool(["oge1","oge2"]))

print(bool(""))

print(bool(False))

print(bool(None))

print(bool(0))

print(bool(()))

print(bool([]))

print(bool({}))
```

```
True

True

True

False

False

False

False

False

False

False
```

```
class sinif():

    def __len__(self):

        return 0



nesne=sinif()
print(bool(nesne))
```

```
False
```

Bir önceki sayfada bool(0) 'ın karşılığının False olduğunu görmüştük.

Aritmetik Operatörler

```
print(5 + 3)
print(5 - 3)
print(5 * 3)
print(12 / 3)
print(5 % 2)
print(5 ** 2)
print(15 // 2)
```

```
8
2
15
4.0
1
25
7
```

```
sayi=5
print(sayi)
```

```
5
```

```
sayi=5
sayi+=3
print(sayi)
```

```
8
```

```
sayi=5
sayi-=3
print(sayi)
```

```
2
```

```
sayi=5
sayi*=3
print(sayi)
```

```
15
```

```
sayi=5
sayi/=3
print(sayi)
```

```
1.6666666666666667
```

```
sayi=5
sayi%=3
print(sayi)
```

```
2
```

```
sayi=5
sayi//=3
print(sayi)
```

```
1
```

```
sayi=5
sayi**=3
print(sayi)
```

```
125
```

```
sayi=5
sayi &= 3
print(sayi)
```

```
1
```

```
sayi=5
sayi|=3
print(sayi)
```

```
7
```

```
sayi=5
sayi^=3
print(sayi)
```

```
6
```

```
sayi=5
sayi>>=3
print(sayi)
```

```
0
```

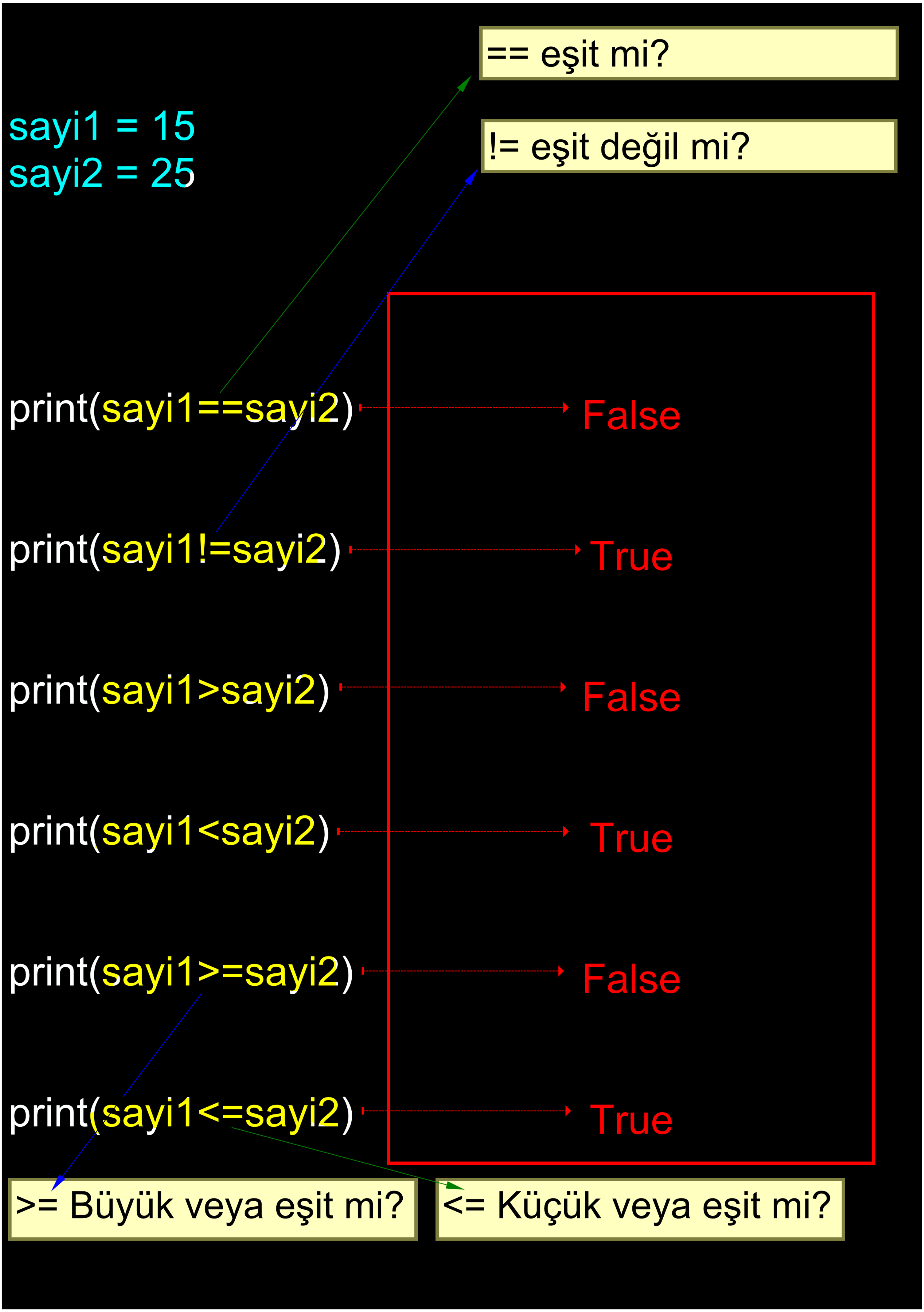
```
sayi=5
sayi<<=3
print(sayi)
```

```
40
```

```
sayi1 = 15
sayi2 = 25
```

```
print(sayi1 + sayi2)  →  40

print(sayi2 - sayi1)  →  10

print(sayi1 * sayi2)  →  375

print(sayi2 / sayi1)  →  1.6666666666666667

print(sayi2 % sayi1)  →  10

print(sayi1**sayi2)   →  25251168294042348861694
                         3359375

print(sayi2//sayi1)   →  1
```

15 üssü 25



== eşit mi?

!= eşit değil mi?

```
sayi1 = 15
sayi2 = 25
```

```
print(sayi1==sayi2)
```

False

```
print(sayi1!=sayi2)
```

True

```
print(sayi1>sayi2)
```

False

```
print(sayi1<sayi2)
```

True

```
print(sayi1>=sayi2)
```

False

```
print(sayi1<=sayi2)
```

True

>= Büyük veya eşit mi?

<= Küçük veya eşit mi?

```
sayi1 = 15
sayi2 = 25


print(sayi1<sayi2 and sayi1>sayi2)

print(sayi1<sayi2 or sayi1>sayi2)

print(not (sayi1<sayi2 and sayi1>sayi2))

print(not  (sayi1<sayi2 or sayi1>sayi2))
```

```
False

True

True

False
```

And : Her iki ifade de doğruysa True döndürür

Or : İfadelerden biri doğruysa True döndürür

not() : Sonucu tersine çevirir, sonuç doğruysa False döndürür

```
alisverisListem =
["yoğurt","ekmek","sucuk","kömür"]

for listele in alisverisListem:
    print(listele)
```

```
yoğurt
ekmek
sucuk
kömür
```

```
alisverisListem = ["yoğurt", "ekmek", "sucuk",
"kömür"]
yeniListem = []

for ara in alisverisListem:
    if "c" in ara:
        yeniListem.append(ara)
print(yeniListem)
```

```
['sucuk']
```

1) Liste oluşturduk

2) Bir de boş liste oluşturduk

3) Döngü oluşturduk ; döngü, listemizi yazdıracak fakat belirlediğimiz kriterleri baz alarak yazdıracak , bu kriter atamasını "ara" şeklinde adlandırdık

4) Kriterimiz c harfi bulundurmalı diye belirledik
5) Kriterimize uyan liste öğelerini, yeniListem'e eklettik ve en son yeniListem'i yazdırdık

```
alisverisListem = ["yoğurt", "ekmek", "sucuk", "kömür"]

yeniListem = [ara for ara in alisverisListem if "ğ" in ara]

print(yeniListem)
```

```
['yoğurt']
```

1) Alışveriş listesi oluşturduk
2) Yeni liste oluşturduk ve içine , ara adında, kriter atayacağımız bir değişken oluşturduk, kriteri ise alisverisListem'in öğelerini döngüye alıp, ğ içeren öğeleri yeniListem'e atadık ve yeniListem'i ekranda gösterdik

tuple : demet

```
demetim = ("nar","çilek","incir")

listele=list(demetim)

listele[1]="armut"

demetim=tuple(listele)

print(demetim)
```

```
('nar', 'armut', 'incir')
```

```
demet_ = ("elma",)
print(type(demet_))

demet__ = ("elma")
print(type(demet__))
```

```
<class 'tuple'>

<class 'str'>
```

```
demet1 = ("a1","a2","a3")
demet2 = ("b1","b2","b3")

demetTopla = demet1 + demet2
print(demetTopla)
```

```
('a1', 'a2', 'a3', 'b1', 'b2', 'b3')
```

```
demet = {"a","b","c"}

for yazdir in demet :
     print(yazdir)
```

```
b
c
a
```

Bir dizindeki öğelere bir dizine başvurarak erişemezsiniz, çünkü demetler sırasız olduğundan öğelerin dizini yoktur. Ancak, bir for döngüsü kullanarak küme öğeleri arasında dolaşabilir veya in anahtar sözcüğünü kullanarak bir kümede belirli bir değerin olup olmadığını sorabilirsiniz.

```
demetim = {"öğe1","öğe2","öğe3"}
demetim.add("öğe4")
print(demetim)
```

```
{'öğe2', 'öğe1', 'öğe4', 'öğe3'}
```

```
demetim = {"öğe1", "öğe2", "öğe3"}
demetim.update(["öğe4","öğe5"])
print(demetim)
```

```
{'öğe3', 'öğe5', 'öğe1', 'öğe4', 'öğe2'}
```

```
bordro = {
      "2019":2575,
      "2020":2650,
      "2021":2790 }

for x,y  in bordro.items():
      print(x,y)
```

```
2019 2575
2020 2650
2021 2790
```

```
bordro = {
      "2019":2575,
      "2020":2650,
      "2021":2790 }

if "2021" in bordro:
      print("2021 yılına ait veri bulunmaktadır")
```

```
2021 yılına ait veri bulunmaktadır
```

```
uyeListesi = {
      "1.Kayit" : "Nazmi Bulut",
      "2.Kayit" : "Damla Özkan",
      "3.Kayit" : "Jale Densiz" }

uyeListesi.pop("3.Kayit")
print(uyeListesi)
```

```
{'1.Kayit': 'Nazmi Bulut', '2.Kayit': 'Damla Özkan'}
```

```
uyeListesi = {
      "1.Kayit" : "Nazmi Bulut",
      "2.Kayit" : "Damla Özkan",
      "3.Kayit" : "Jale Densiz" }

uyeListesi.popitem()
print(uyeListesi)
```

```
{'1.Kayit': 'Nazmi Bulut', '2.Kayit': 'Damla Özkan'}
```

```
uyeListesi = {
      "1.Kayit" : "Nazmi Bulut",
      "2.Kayit" : "Damla Özkan",
      "3.Kayit" : "Jale Densiz" }

del uyeListesi["2.Kayit"]
print(uyeListesi)
```

```
{'1.Kayit': 'Nazmi
Bulut', '3.Kayit':
'Jale Densiz'}
```

```
aileUyeleri = {
      "cocuk1" : { "Ad Soyad" : "Nazmi Bulut" },
      "cocuk2" : { "Ad Soyad" : "Damla Özkan" },
      "anne" : { "Ad Soyad" : "Jale Densiz"},
      "baba" : { "Ad Soyad" : "Muhittin Karapınar" }
      }
print(aileUyeleri)
```

```
{'cocuk1': {'Ad Soyad': 'Nazmi Bulut'}, 'cocuk2':
{'Ad Soyad': 'Damla Özkan'}, 'anne': {'Ad Soyad':
'Jale Hancı'}, 'baba': {'Ad Soyad': 'Muhittin
Karapınar'}}
```

```
sayi1=55
sayi2=105

if sayi1<sayi2:
      print("sayi2 büyüktür sayi1'den")
```

```
sayi2 büyüktür sayi1'den
```

```
sayi1=55
sayi2=105

print("sayi1") if sayi1>sayi2 else print("sayi2")
```

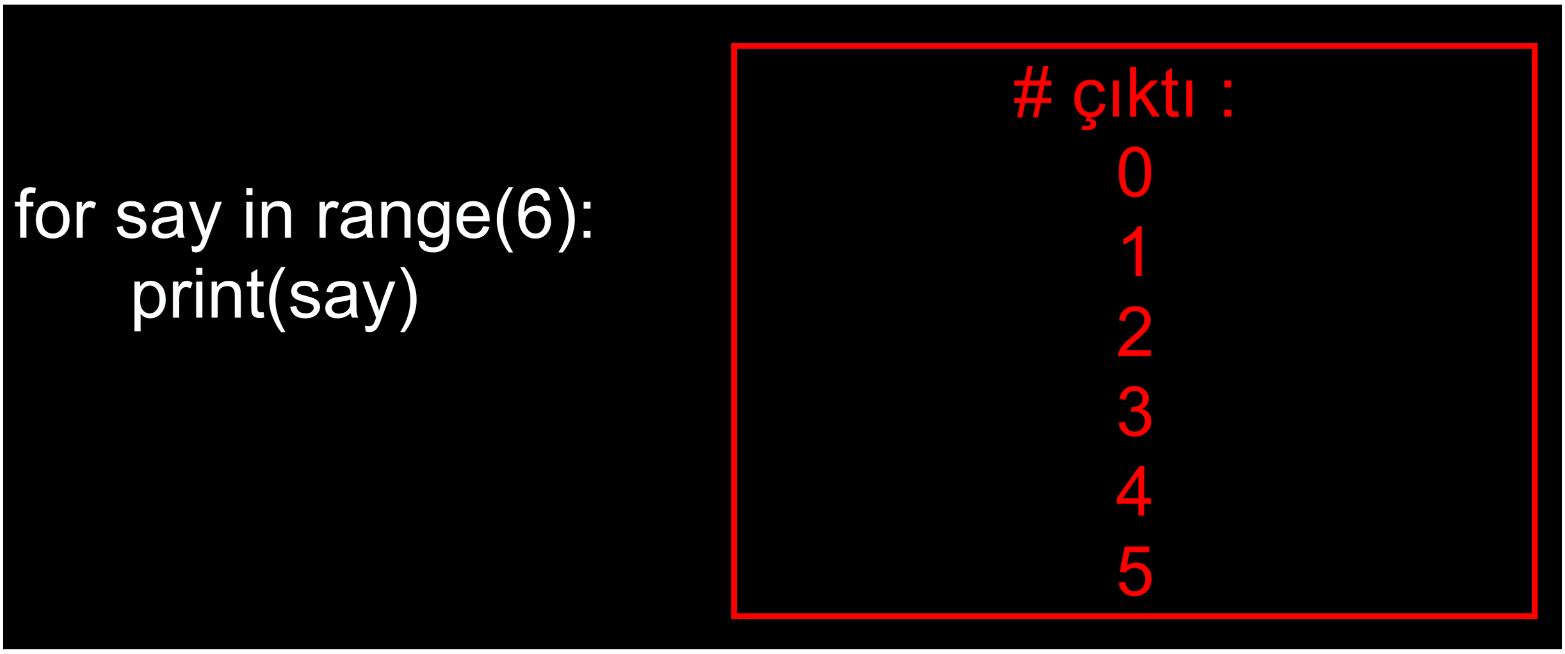
```
sayi2
```

```
sayi1=300
sayi2=300

print("sayi1") if sayi1>sayi2 else print("=") if
sayi1==sayi2 else print("sayi2")
```

```
=
```

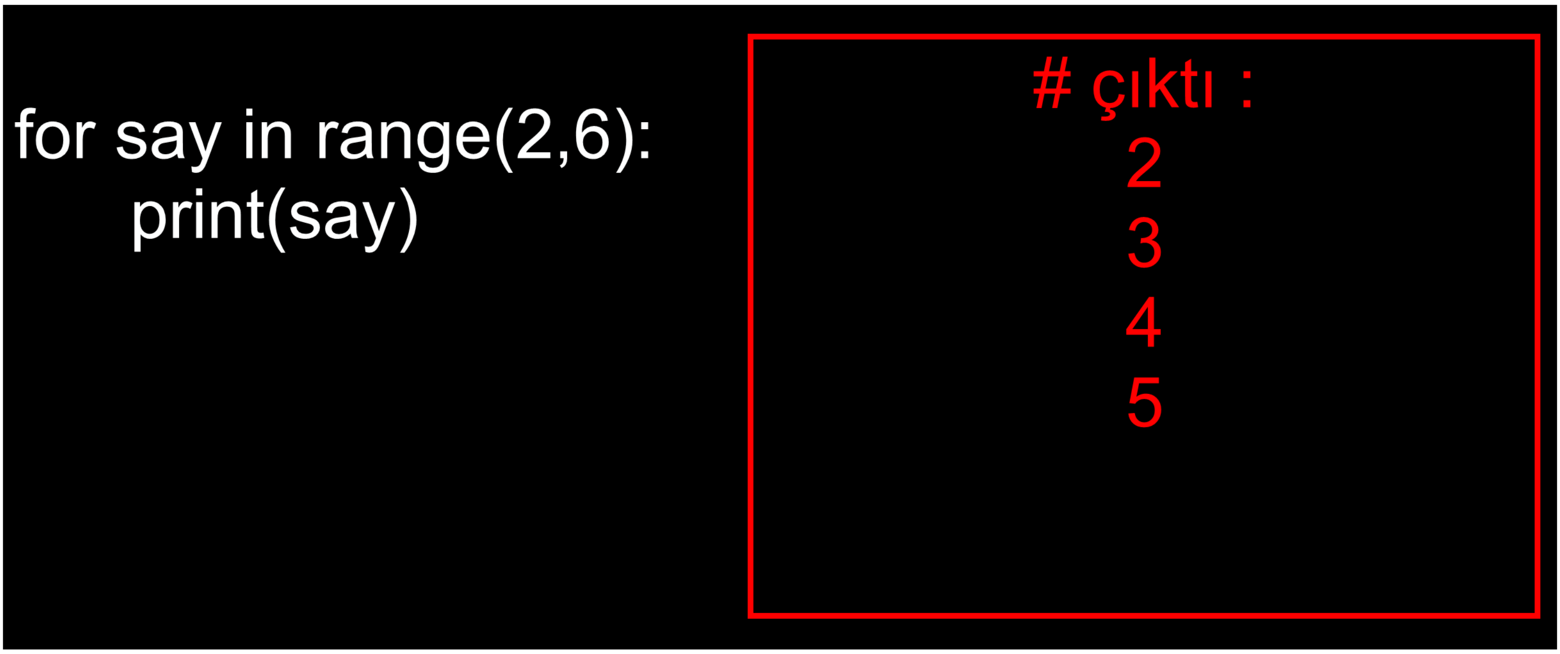
```
sayi1=100
sayi2=200
sayi3=300

if sayi2>sayi1 or sayi1>sayi3:
      print("Koşullardan en az biri doğru")

else: print("Hiçbir koşul doğru değil")
```

```
1
2
3
4
5
Sayi değeri artık 6'dan küçük değil
```



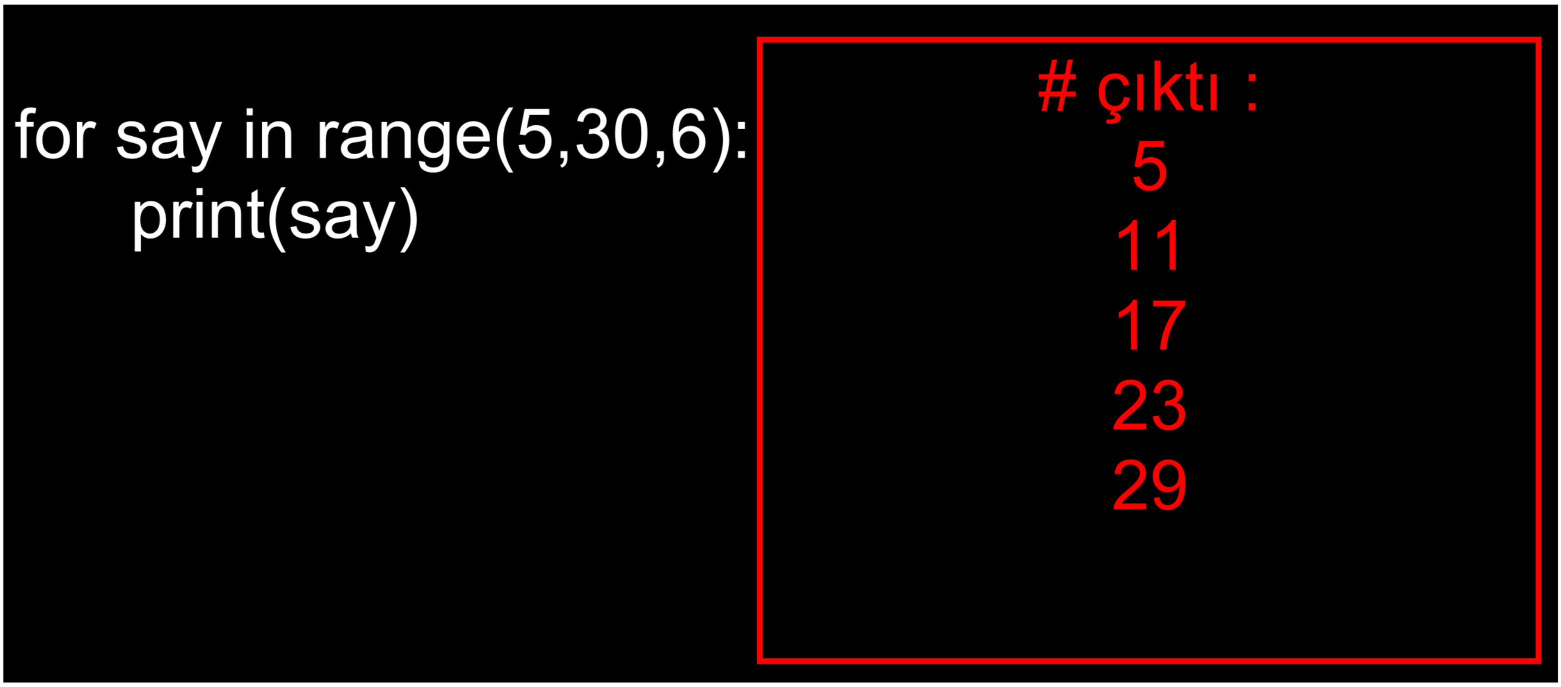
```
for say in range(6):
      print(say)
```

```
# çıktı :
0
1
2
3
4
5
```



```
for say in range(2,6):
      print(say)
```

```
# çıktı :
2
3
4
5
```



```
for say in range(5,30,6):
      print(say)
```

```
# çıktı :
5
11
17
23
29
```

```
liste1 = ["sarı","arı","farı"]
liste2 = ["limon","marul"]



for liste1kesisim2 in liste1:
    for liste2kesisim1 in liste2:
        print(liste1kesisim2,liste2kesisim1)
```

```
# çıktı :
sarı limon
sarı marul
arı limon
arı marul
farı limon
farı marul
```

```
def fonksiyonum(fonksiyon_ismi):
    print(fonksiyon_ismi+ " Bilgisi")

fonksiyonum("Email")
fonksiyonum("Telefon")
fonksiyonum("Ev Adresi")
```

```
# çıktı :
Email Bilgisi
Telefon Bilgisi
Ev Adresi Bilgisi
```

```
def fonksiyonum(isim,soyisim):
    print(isim+ " " + soyisim)

fonksiyonum("Yağız" ,"Ertunç")
```

```
# çıktı : Yağız Ertunç
```

```
def fonksiyon(*argumanlar):
    print("En genç üye : " + argumanlar[2])
    # 2.indexi yani 3. kişiyi getirir

fonksiyon("Nilay","Oğuz","Mahmut","Ertuğrul","Naz
")
```

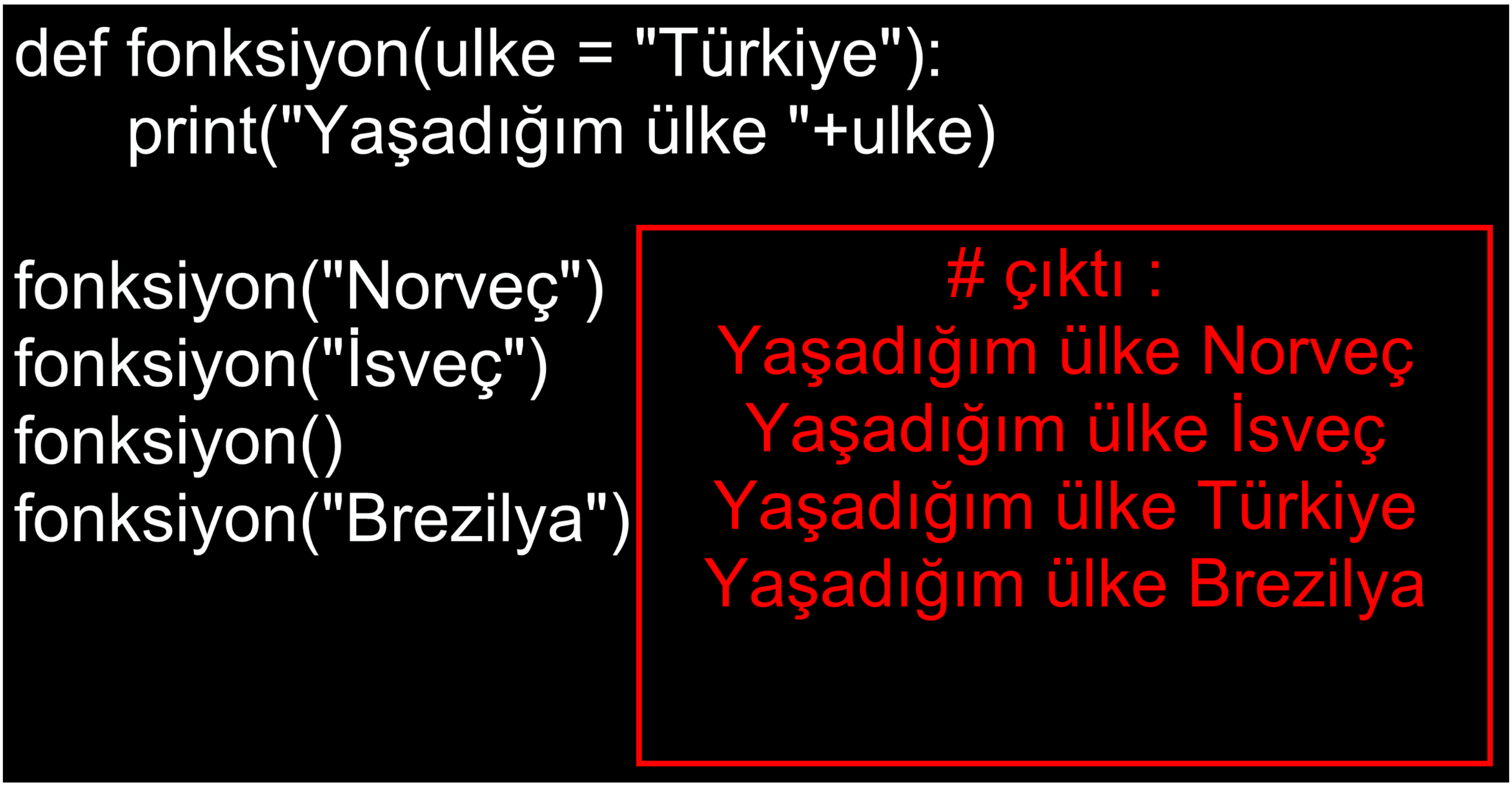
```
# çıktı : En genç üye : Mahmut
```

```
def kisiler(kisi1,kisi2,kisi4,kisi3):
    print("En yaşlı üye "+ kisi2)

kisiler(
    kisi1="Diyar",
    kisi2="Merve",
    kisi3="Vefa",
    kisi4="Numan")
```

```
# çıktı : En yaşlı
üye Merve
```

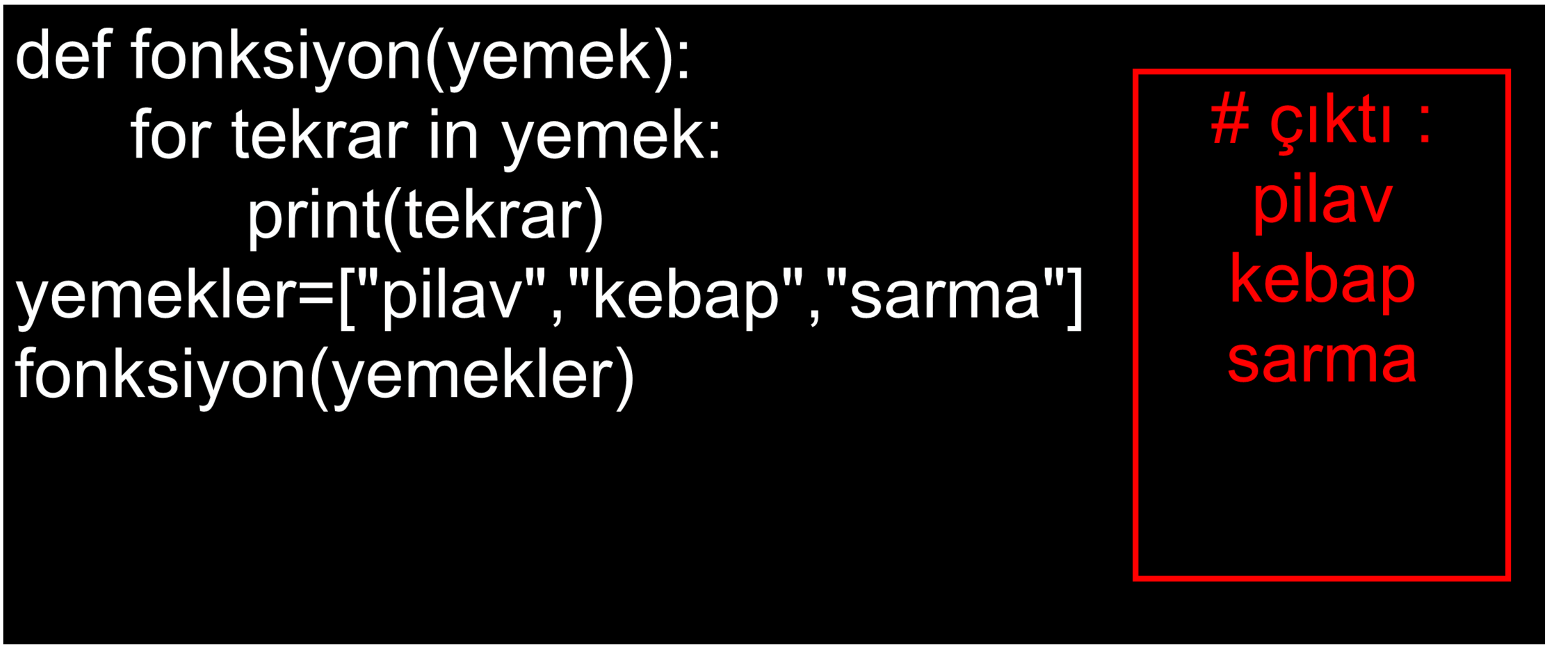
```
def fonksiyonum(**kisi):
    print("Adınız : " + kisi["adiniz"] + "\n" +
"Soyadınız : " + kisi["soyad"])

fonksiyonum(adiniz="Süleyman",soyad="Çelik")
```

```
# çıktı :
Adınız : Süleyman
Soyadınız : Çelik
```



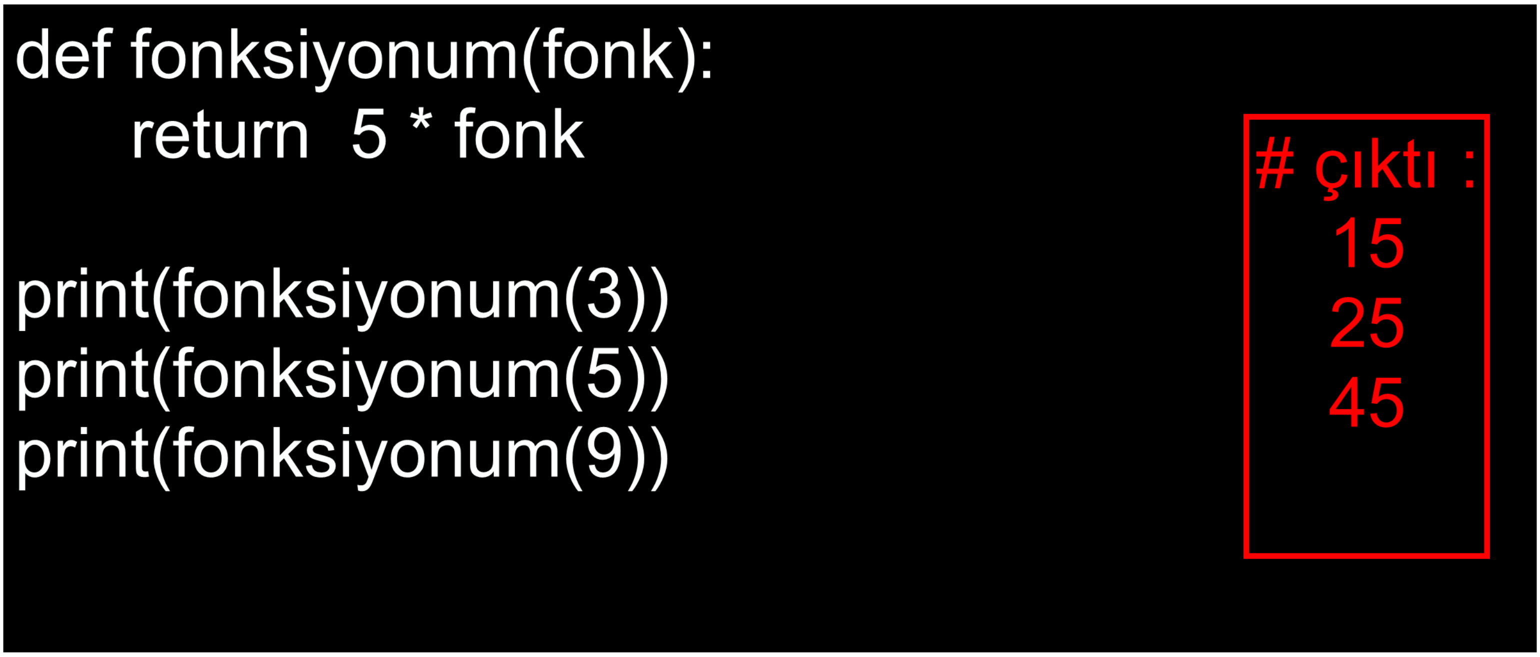
```
def fonksiyon(ulke = "Türkiye"):
      print("Yaşadığım ülke "+ulke)

fonksiyon("Norveç")
fonksiyon("İsveç")
fonksiyon()
fonksiyon("Brezilya")
```

```
# çıktı :
Yaşadığım ülke Norveç
Yaşadığım ülke İsveç
Yaşadığım ülke Türkiye
Yaşadığım ülke Brezilya
```



```
def fonksiyon(yemek):
      for tekrar in yemek:
            print(tekrar)
yemekler=["pilav","kebap","sarma"]
fonksiyon(yemekler)
```

```
# çıktı :
pilav
kebap
sarma
```



```
def fonksiyonum(fonk):
      return  5 * fonk

print(fonksiyonum(3))
print(fonksiyonum(5))
print(fonksiyonum(9))
```

```
# çıktı :
15
25
45
```

```
def yineleme(sayi):

    if(sayi>0):
        result=sayi+yineleme(sayi-1)
        print(result)
    else:
        result=0
    return result

print("\nSonuçlar")
yineleme(6)
```

```
Sonuçlar
1
3
6
10
15
21
```

```
sayi = lambda x:x+10
print(sayi(30))
```

```
40
```

```
sayi = lambda x,y:x+y
print(sayi(10,20))
```

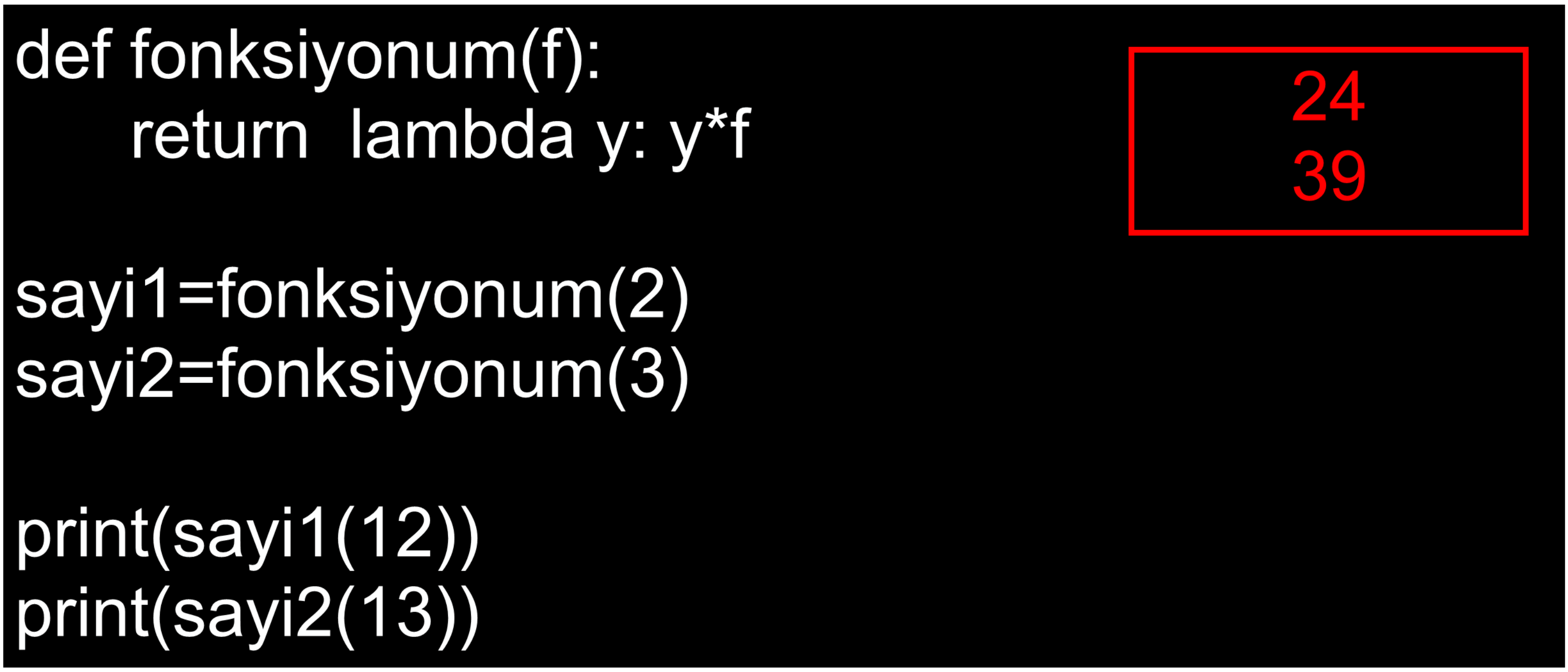
```
30
```

```
sayi = lambda x,y,z : x*y*z
print(sayi(3,2,1))
```

```
6
```

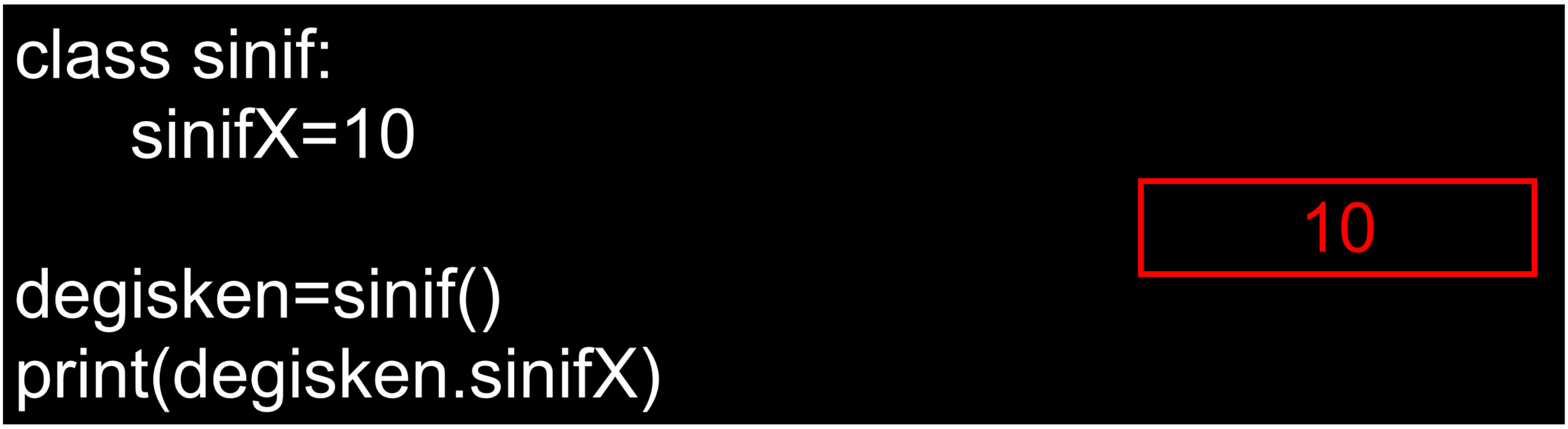
```
def fonksiyonum(f):
    return  lambda x: x*f

sayi=fonksiyonum(2)
print(sayi(12))
```

```
24
```



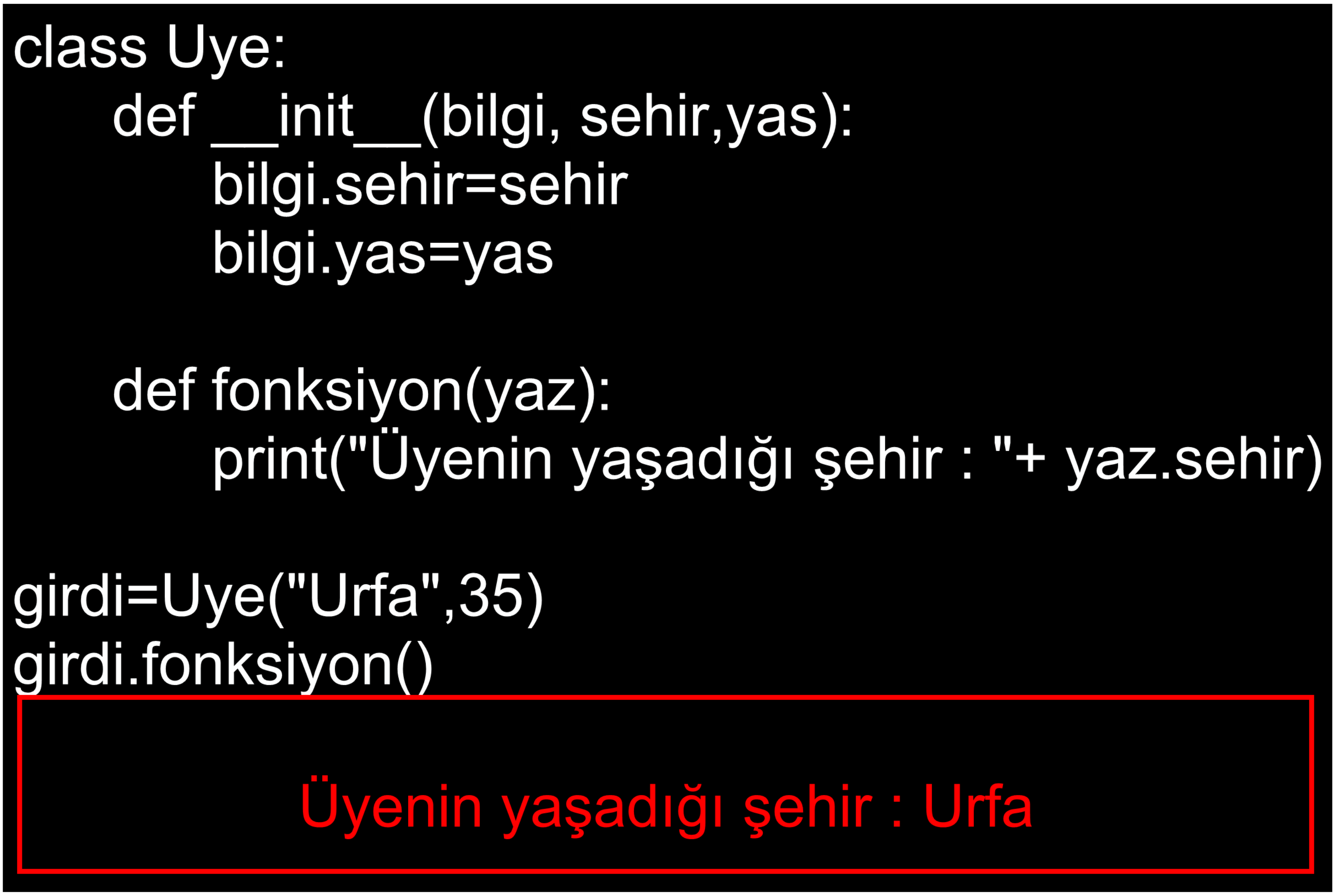
```
def fonksiyonum(f):
    return  lambda y: y*f

sayi1=fonksiyonum(2)
sayi2=fonksiyonum(3)

print(sayi1(12))
print(sayi2(13))
```

```
24
39
```



```
class sinif:
    sinifX=10

degisken=sinif()
print(degisken.sinifX)
```

```
10
```

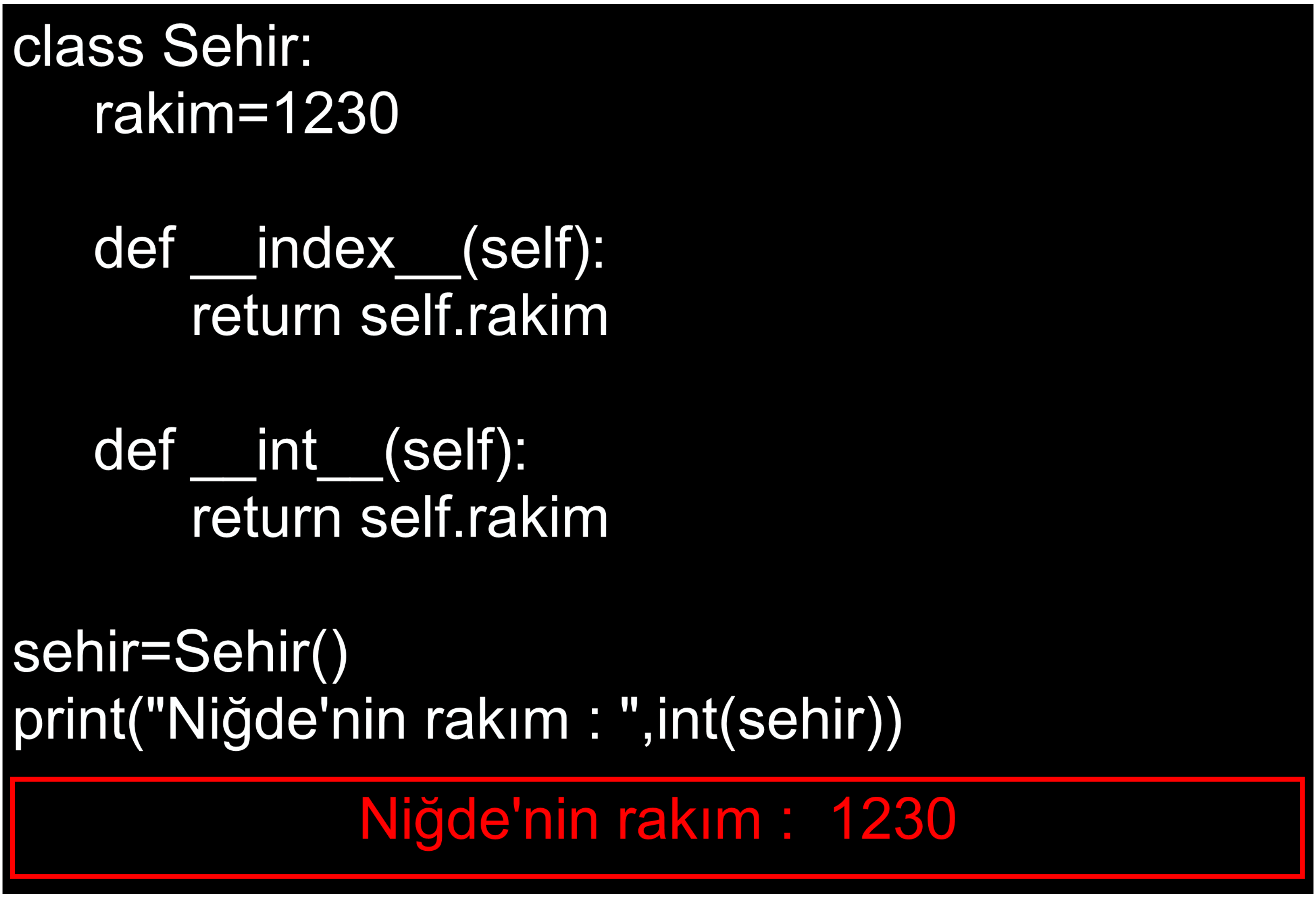


```
class Uye:
    def __init__(bilgi, sehir,yas):
        bilgi.sehir=sehir
        bilgi.yas=yas

    def fonksiyon(yaz):
        print("Üyenin yaşadığı şehir : "+ yaz.sehir)

girdi=Uye("Urfa",35)
girdi.fonksiyon()
```

```
Üyenin yaşadığı şehir : Urfa
```

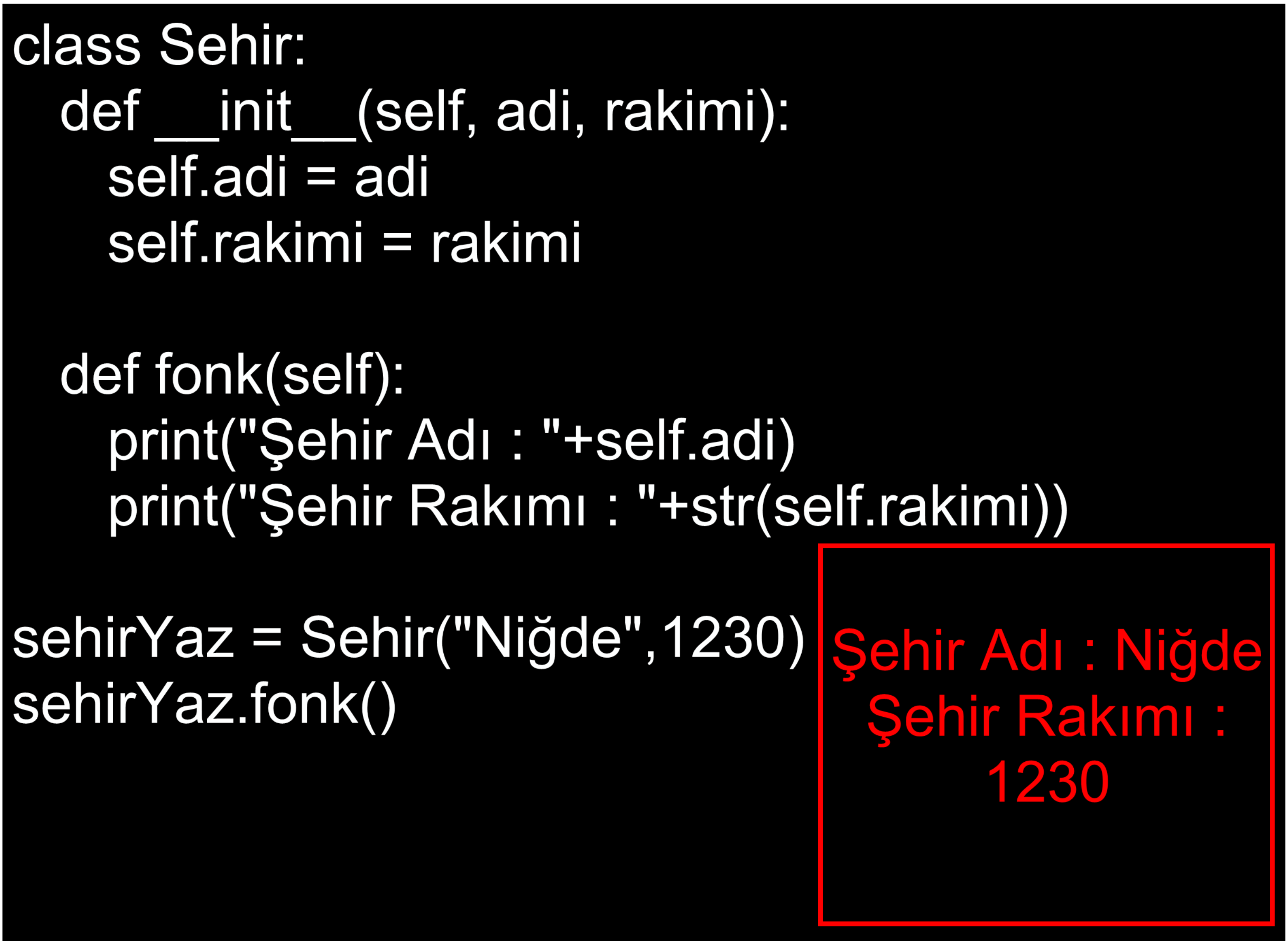
```
class calisan:
    def __init__(self,ad_soyad ,departman):
        self.ad_soyad = ad_soyad
        self.departman=departman

    def fonksiyon(self):
        print("Çalışanın Adı ve Soyadı: "+
self.ad_soyad + "\n" + "Departmanı :
"+self.departman)

bilgi=calisan("Aykut Hasanoğlu","IT""\n")
bilgi.fonksiyon()

bilgi=calisan("Nazan İnce","IK""\n")
bilgi.fonksiyon()
```

```
Çalışanın Adı ve Soyadı: Aykut Hasanoğlu
Departmanı : IT

Çalışanın Adı ve Soyadı: Nazan İnce
Departmanı : IK
```



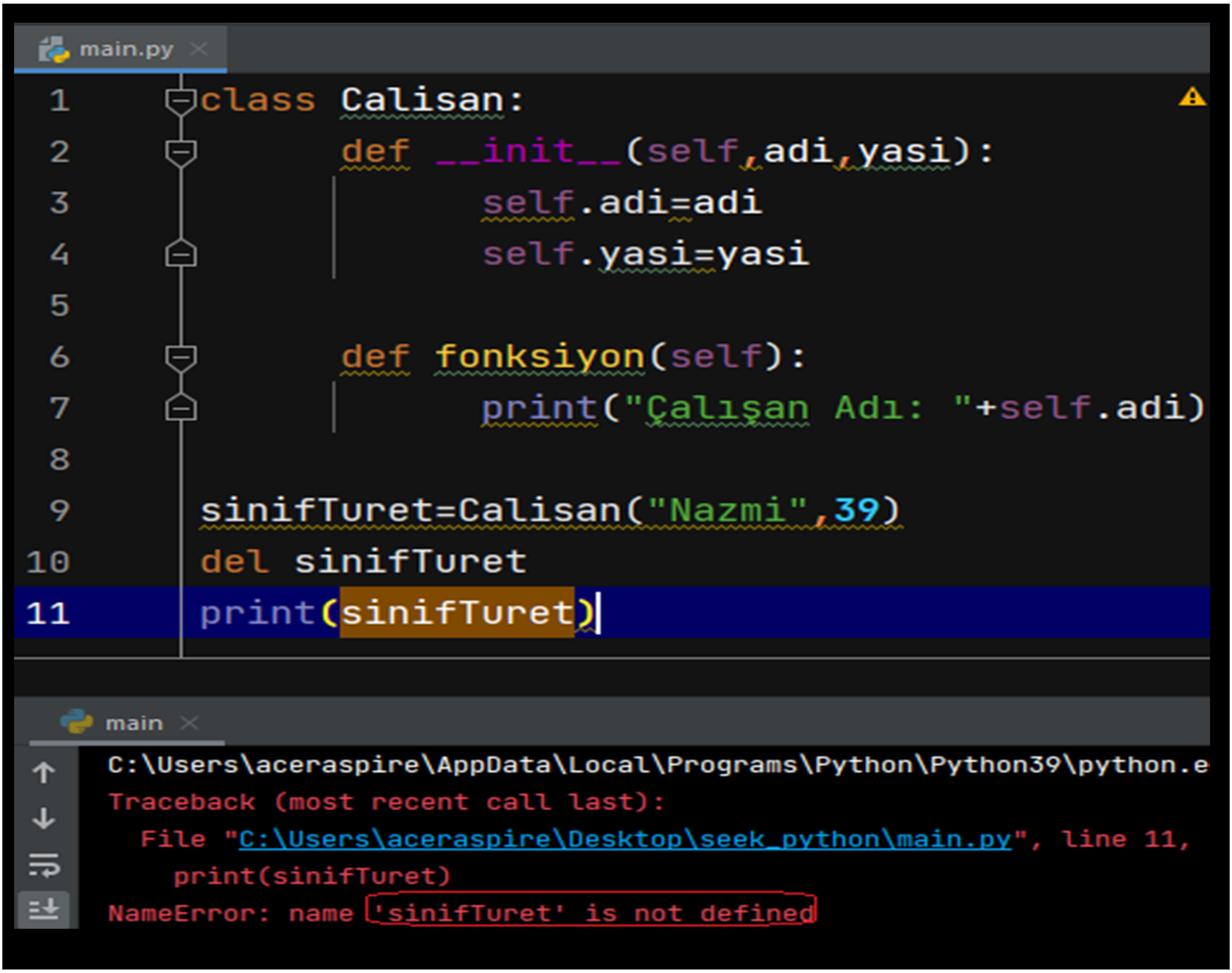
```
class Sehir:
    rakim=1230

    def __index__(self):
        return self.rakim

    def __int__(self):
        return self.rakim

sehir=Sehir()
print("Niğde'nin rakım : ",int(sehir))
```

```
Niğde'nin rakım :  1230
```



```
class Sehir:
  def __init__(self, adi, rakimi):
    self.adi = adi
    self.rakimi = rakimi

  def fonk(self):
    print("Şehir Adı : "+self.adi)
    print("Şehir Rakımı : "+str(self.rakimi))

sehirYaz = Sehir("Niğde",1230)
sehirYaz.fonk()
```

```
Şehir Adı : Niğde
Şehir Rakımı :
1230
```



```python
class Calisan:
    def __init__(self,adi,yasi):
        self.adi=adi
        self.yasi=yasi

    def fonksiyon(self):
        print("Çalışan Adı: "+self.adi)

sinifTuret=Calisan("Nazmi",39)
del sinifTuret
print(sinifTuret)
```

```
C:\Users\aceraspire\AppData\Local\Programs\Python\Python39\python.e
Traceback (most recent call last):
  File "C:\Users\aceraspire\Desktop\seek_python\main.py", line 11,
    print(sinifTuret)
NameError: name 'sinifTuret' is not defined
```

```python
class Kisi:
    def __init__(self,Kadi,Ksoyadi):
        self.Kadi=Kadi
        self.Ksoyadi=Ksoyadi

    def fonksiyon(self):
        print(self.Kadi, self.Ksoyadi)

yazdir=Kisi("Cüneyt","Çalışkan")
yazdir.fonksiyon()
```

```
Cüneyt Çalışkan
```

```
class Kisi:
    def __init__(self,Kadi,Ksoyadi):
        self.Kadi=Kadi
        self.Ksoyadi=Ksoyadi

    def fonksiyon(self):
        print(self.Kadi, self.Ksoyadi)
class Calisan(Kisi):
    pass
yazdir=Kisi("Meltem","Candan")
yazdir.fonksiyon()
```

```
Meltem
Candan
```

```
class Katilimci:
    def __init__(self,katilimNo,katilimUcret):
        self.katilimNo=katilimNo
        self.katilimUcret=katilimUcret

    def fonksiyon(self):
        print(self.katilimNo, self.katilimUcret)

class Seyirci(Katilimci):
    def __init__(self,katilimNo,katilimUcret):

Katilimci.__init__(self,katilimNo,katilimUcret)

yazdir=Seyirci("Gürkan",30)
yazdir.fonksiyon()
```

```
Gürkan 30
```

```
class Katilimci:

	def __init__(self,katilimNo,katilimUcret):
		self.katilimNo=katilimNo
		self.katilimUcret=katilimUcret


	def fonksiyon(self):
print(int(self.katilimNo),int(self.katilimUcret))




class Seyirci(Katilimci):
	def __init__(self,katilimNo,katilimUcret):
		super().__init__(katilimNo,katilimUcret)



yazdir=Seyirci(30,250)
yazdir.fonksiyon()
```

```
30 250
```

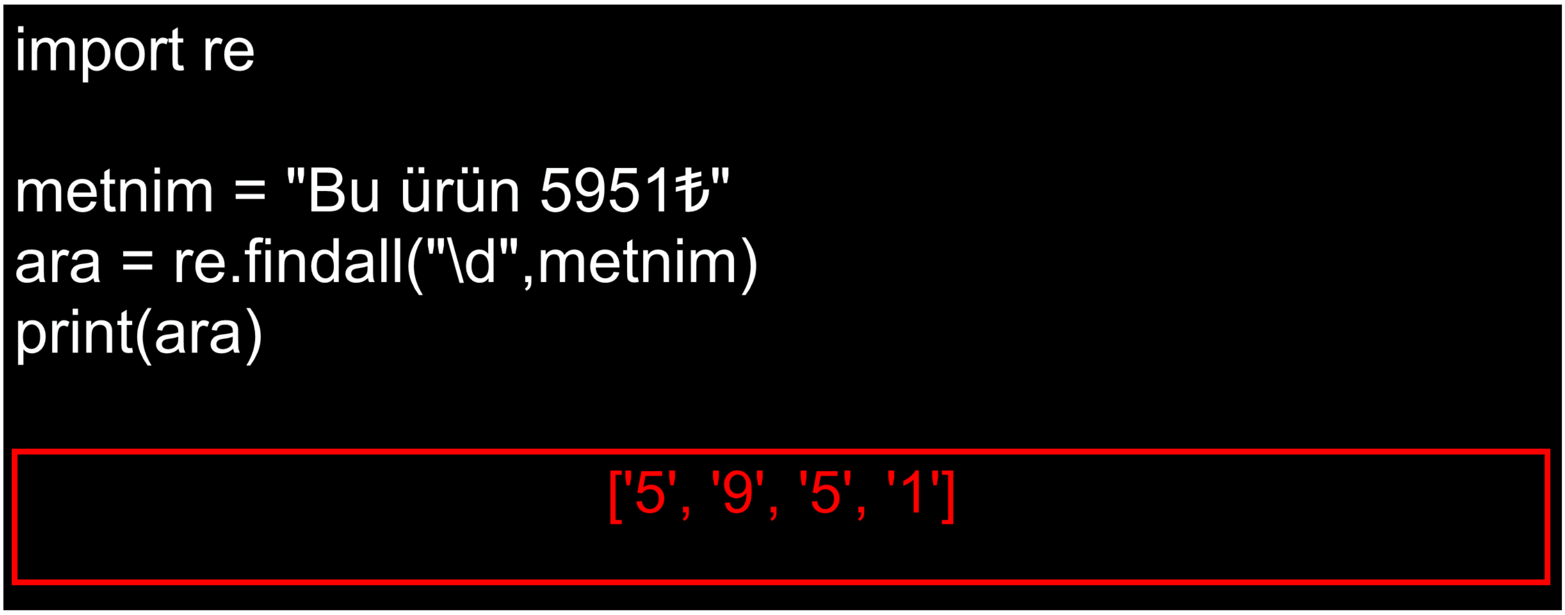
```
class Katilimci:

    def __init__(self,katilimNo,katilimUcret):
        self.katilimNo=katilimNo
        self.katilimUcret=katilimUcret

    def fonksiyon(self):
print(int(self.katilimNo),int(self.katilimUcret))



class Seyirci(Katilimci):

    def __init__(self,katilimNo,katilimUcret):
        super().__init__(katilimNo,katilimUcret)
        self.filmYili=2019


yazdir=Seyirci(30,250)
print(yazdir.filmYili)
```

```
2019
```

```
class Seyirci:
    def
__init__(self,metin1,koltukNo,metin2,filmAdi,metin3,girisUcreti):
        self.metin1=metin1
        self.koltukNo=koltukNo
        self.metin2=metin2
        self.filmAdi=filmAdi
        self.metin3=metin3
        self.girisUcreti=girisUcreti


    def fonksiyon(self):
        print(str(self.metin1),
            int(self.koltukNo),
            str(self.metin2),
            str(self.filmAdi),
            str(self.metin3),
            int(self.girisUcreti))



class yeniSinifTuret(Seyirci):
    def
__init__(self,metin1,koltukNo,metin2,filmAdi,metin3,girisUcreti):

super().__init__(metin1,koltukNo,metin2,filmAdi,metin3,girisUcreti
)

yazdir=yeniSinifTuret("Koltuk No :",56,"& Film Adı :","Deneme","&
Giriş Ücreti :",51)
yazdir.fonksiyon()
```

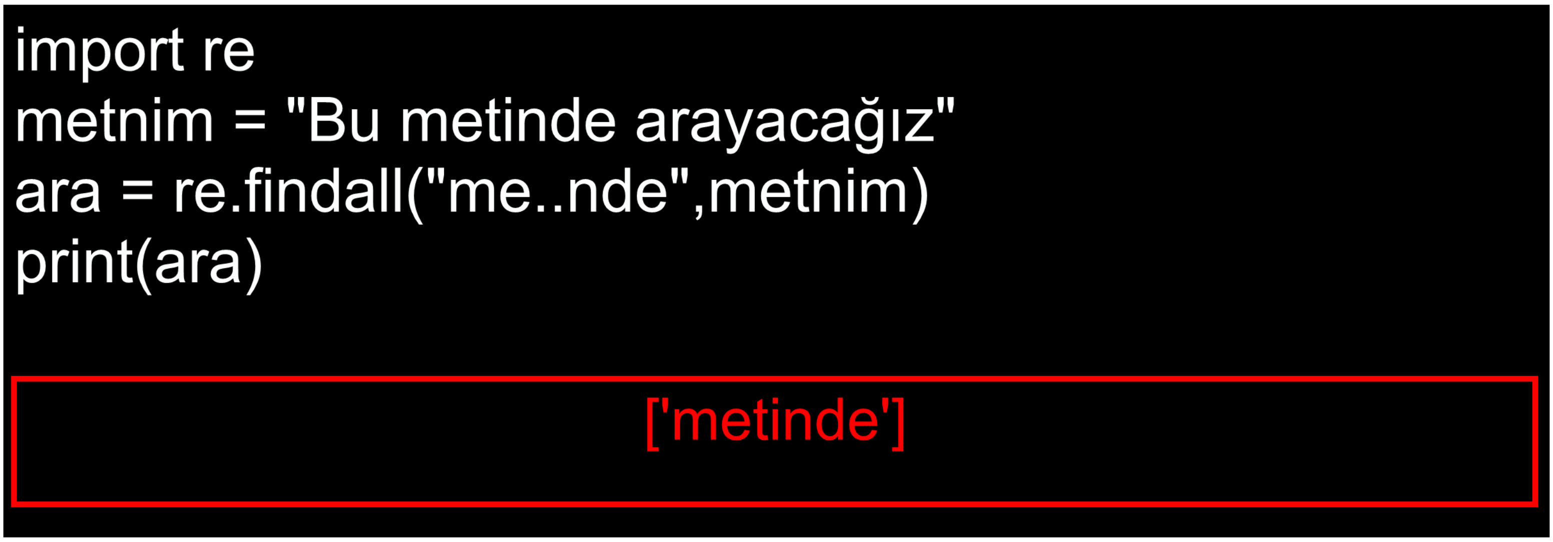
Boşluk bırakılmadı

```
Koltuk No : 56 & Film Adı : Deneme & Giriş
Ücreti : 51
```



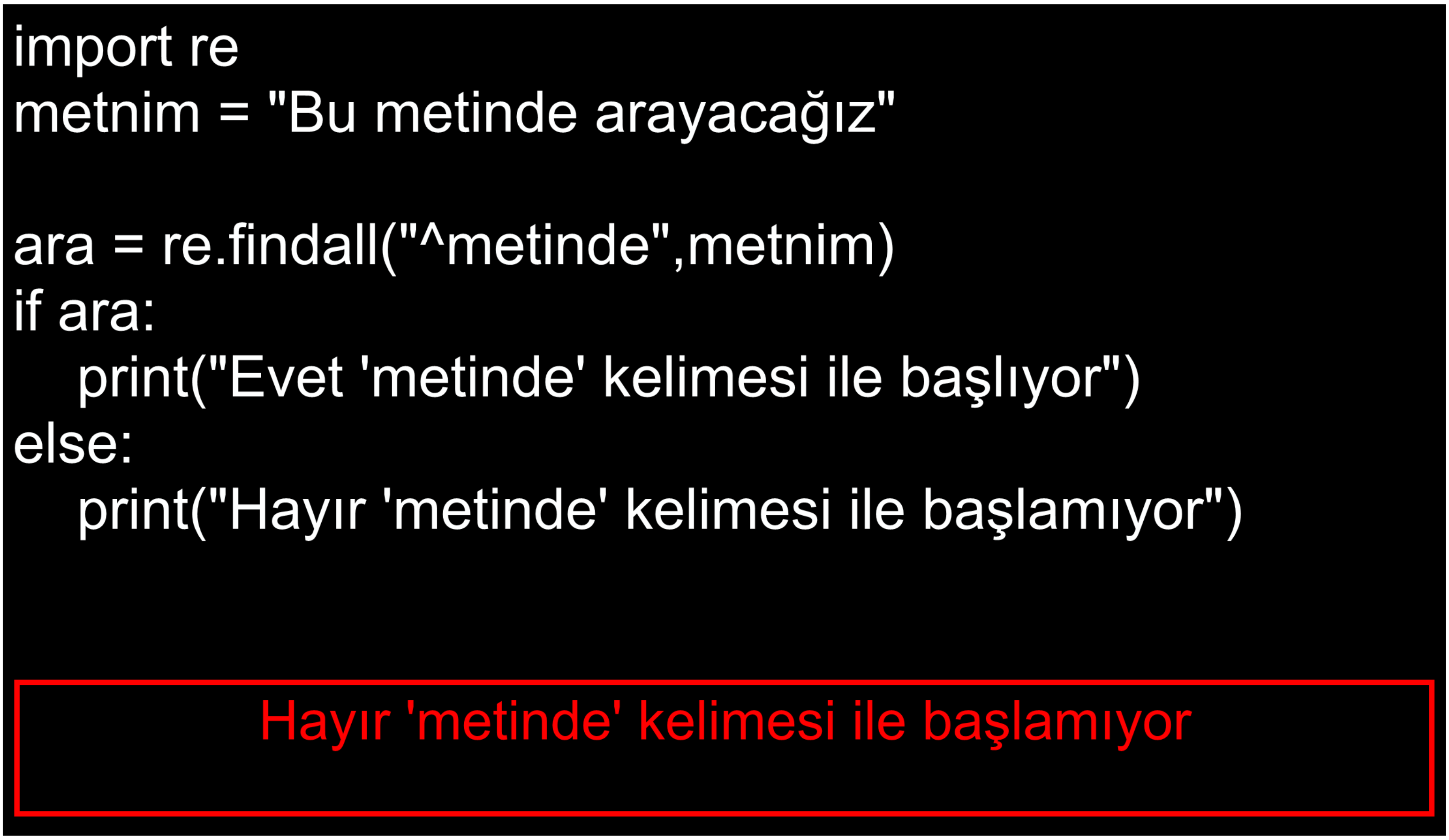
```
import re

metnim = "Bu ürün 5951₺"
ara = re.findall("\d",metnim)
print(ara)
```

```
['5', '9', '5', '1']
```



```
import re
metnim = "Bu metinde arayacağız"
ara = re.findall("me..nde",metnim)
print(ara)
```

```
['metinde']
```



```
import re
metnim = "Bu metinde arayacağız"

ara = re.findall("^metinde",metnim)
if ara:
    print("Evet 'metinde' kelimesi ile başlıyor")
else:
    print("Hayır 'metinde' kelimesi ile başlamıyor")
```

```
Hayır 'metinde' kelimesi ile başlamıyor
```

```
import re
metnim = "Bu metinde arayacağız"

ara = re.findall("arayacağız$",metnim)
if ara:
    print("Evet 'arayacağız' kelimesi ile bitiyor")
else:
    print("Hayır 'arayacağız' kelimesi ile bitmiyor")
```

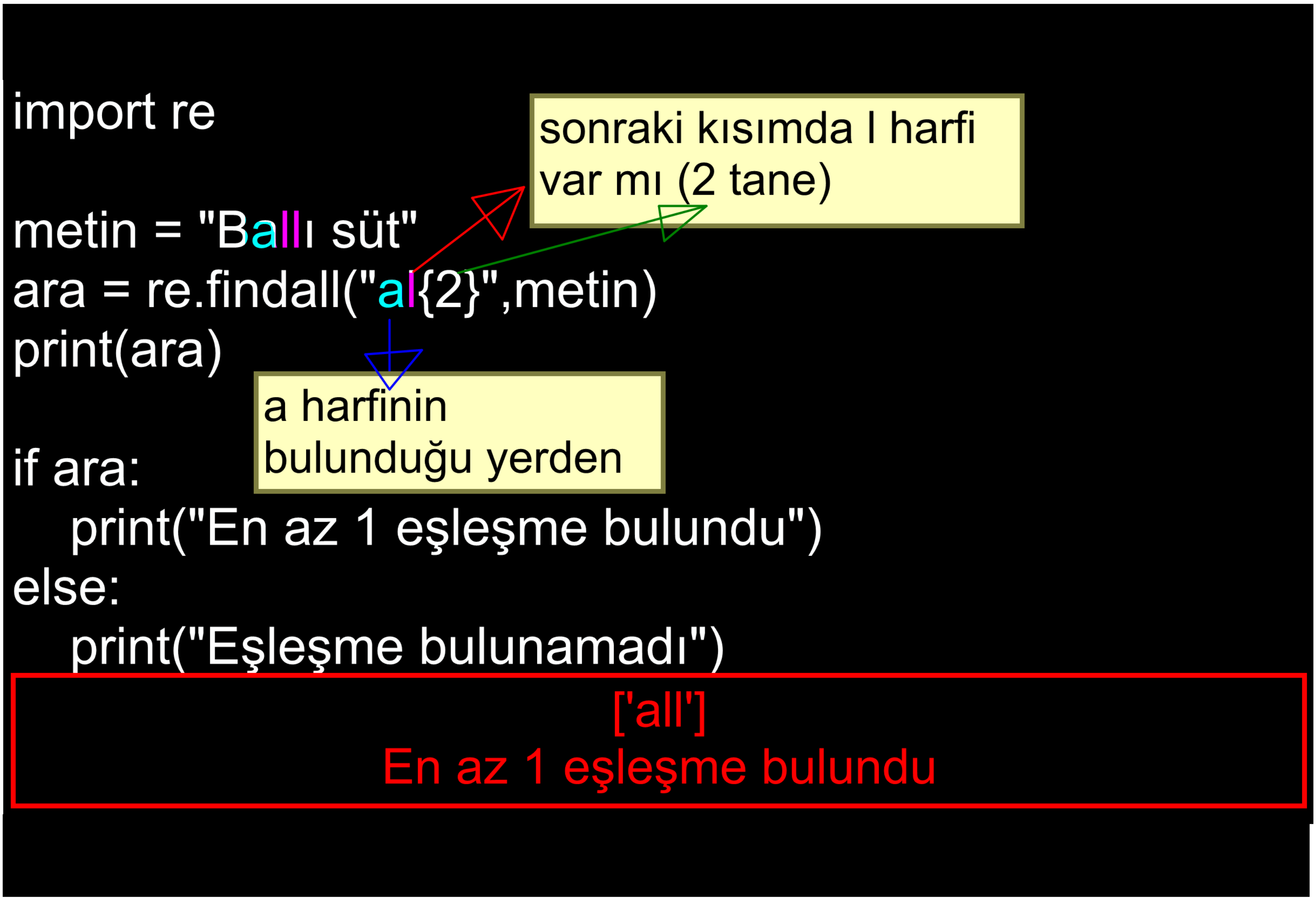
```
Evet 'arayacağız' kelimesi ile bitiyor
```

```
import re
metnim = "Bu metinde arayacağız"

ara = re.findall("ind*",metnim)
if ara:
    print("Evet, en az 1 eşleşme bulundu")
else:
    print("Hayır, eşleşme bulunamadı")
```

```
Evet, en az 1 eşleşme bulundu
```

```
import re
metnimiz = "Yağmurun en fazla yağdığı ilimiz Rize'dir"
ara = re.findall("ağmu",metnimiz)
print(ara)
if ara:
    print("Evet, en az 1 eşleşme bulundu")
else:
    print("Hayır,eşleşme bulunamadı")
```

```
['ağmu']
Evet, en az 1 eşleşme bulundu
```

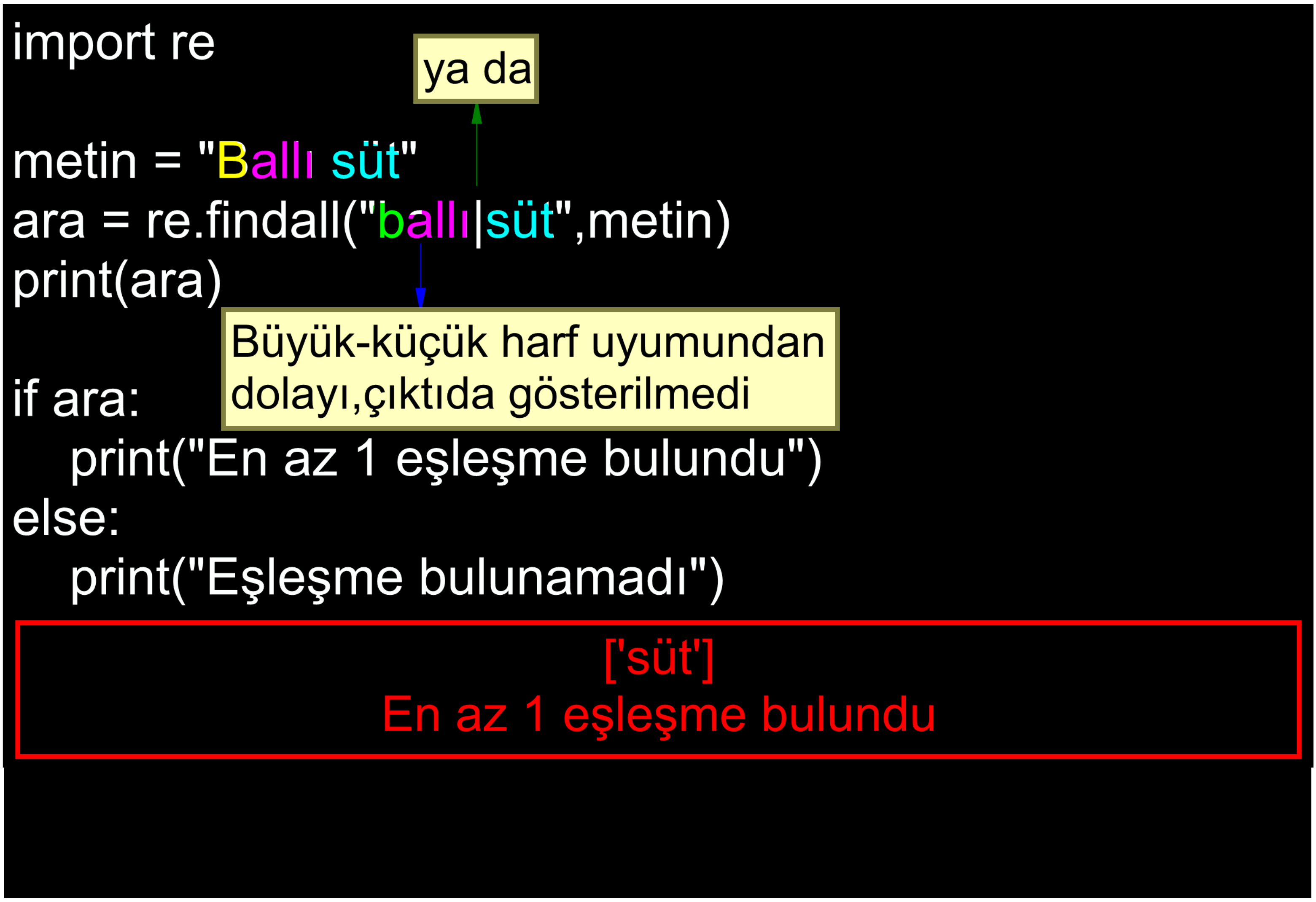


```
import re

metin = "Ballı süt"
ara = re.findall("al{2}",metin)
print(ara)

if ara:
    print("En az 1 eşleşme bulundu")
else:
    print("Eşleşme bulunamadı")
```

sonraki kısımda l harfi var mı (2 tane)

a harfinin bulunduğu yerden

```
['all']
En az 1 eşleşme bulundu
```



```
import re

metin = "Ballı süt"
ara = re.findall("ballı|süt",metin)
print(ara)

if ara:
    print("En az 1 eşleşme bulundu")
else:
    print("Eşleşme bulunamadı")
```

ya da

Büyük-küçük harf uyumundan dolayı,çıktıda gösterilmedi

```
['süt']
En az 1 eşleşme bulundu
```

```
import re

metin = "Bu örnekte '\A'yı inceleyeceğiz"
ara = re.findall("\ABu",metin)
print(ara)

if ara:
    print("Eşleşme bulundu")
else:
    print("Eşleşme bulunamadı")
```

\A
Belirtilen karakterler dizenin başındaysa bir eşleşme döndürür

```
['Bu']
Eşleşme bulundu
```

```
import re

metin = "'\b' : Kelimelerden herhangi birinin,başında
olup olmadığını kontrol eder"

ara = re.findall(r"\bbaş",metin)
print(ara)

if ara:
    print("En az 1 eşleşme bulundu")
else:
    print("Eşleşme bulunamadı")
```

\b
Belirtilen karakterlerin bir kelimenin başında veya sonunda olduğu bir eşleşme döndürür

```
['baş']
En az 1 eşleşme bulundu
```

```
import re

metnim = "Yağmur yağıyor"

bul = re.findall(r"ıyor\b",metnim)
print(bul)

if bul:
    print("En az 1 eşleşme bulundu")
else:
    print("Eşleşme bulunamadı")
```

\b
Belirtilen karakterlerin bir kelimenin başında veya sonunda olduğu bir eşleşme döndürür

```
['ıyor']
En az 1 eşleşme bulundu
```

```
import re

metnimiz = "Bu örnekte,önceki örnekte gördüğümüze
benzer bir yapı"

kontrol = re.findall(r"\Brnekte",metnimiz)

print(kontrol)

if kontrol:
    print("En az 1 eşleşme bulundu")
else:
    print("Eşleşme bulunamadı")
```

\B
Belirtilen karakterlerin bulunduğu, ancak bir kelimenin başında (veya sonunda) OLMAYAN bir eşleşme döndürür

```
['rnekte', 'rnekte']
En az 1 eşleşme bulundu
```

```
import re
metnimiz = "Bu örnekte,bir önceki örnekte gördüğümüze benzer yapıda"
kontrol = re.findall(r"rnekte\B",metnimiz)
print(kontrol)

if kontrol:
    print("En az 1 eşleşme bulundu")
else:
    print("Eşleşme bulunamadı")
```

\B
Belirtilen karakterlerin bulunduğu, ancak bir kelimenin başında (veya sonunda) OLMAYAN bir eşleşme döndürür

```
[]
Eşleşme bulunamadı
```

```
import re
metin = "Bugün 29 Temmuz 2021"
kontrol = re.findall("\d",metin)
print(kontrol)
if kontrol:
    print("En az 1 eşleşme bulundu")
else:
    print("Hiç eşleşme bulunamadı")
```

\d
Dizenin rakamlar içerdiği (0-9 arası sayılar) bir eşleşme döndürür

```
['2', '9', '2', '0', '2', '1']
En az 1 eşleşme bulundu
```

```
import re
metin="Ayrıştır"
fonksiyon=re.findall("\D",metin)
print(fonksiyon)
```

\D
Dizenin rakam İÇERMEDİĞİ bir eşleşme döndürür

```
['A', 'y', 'r', 'ı', 'ş', 't', 'ı', 'r']
```

```
import re
metin="Burada yağmur yağıyor"

fonksiyon=re.findall("\s",metin)
print(fonksiyon)

if fonksiyon:
    print("Evet, minimum 1 eşleşme bulundu")

else:
    print("Hayır, hiç eşleşme bulunamadı")
```

2 boşluk kullanıldı

```
[' ', ' ']
Evet, minimum 1 eşleşme bulundu
```

```
import re
metnim="Yeni bir örnek"


fonk=re.findall("\S",metnim)
print(fonk)



if fonk:
    print("Evet,minimum 1 eşleşme bulundu")
else:
    print("Hayır,hiç eşleşme bulunamadı")
```

\S
Boşluk karakterini görmezden gelir ve dizedeki tüm karakterleri çıktı olarak verir

```
['Y', 'e', 'n', 'i', 'b', 'i', 'r', 'ö', 'r', 'n', 'e', 'k']
```

```
import re
metin = "Yen! 1 *rnek"

fonksiyonum=re.findall("\w",metin)
print(fonksiyonum)

if fonksiyonum:
    print("Evet,minimum 1 sonuç bulundu")
else:
    print("Hayır,hiç eşleşme bulunamadı")
```

\w
a-z aralığındaki
0-9 karakterleri
_ çıktı olarak
verir
(Alt çizgi)

```
['Y', 'e', 'n', '1', 'r', 'n', 'e', 'k']
Evet,minimum 1 sonuç bulundu
```

```
import re
degisken="Yen1 b!r #rnek"


fonksiyon=re.findall("\W",degisken)
print(fonksiyon)


if fonksiyon:
    print("Evet,en az 1 eşleşme bulundu")

else:
    print("Hayır,hiç eşleşme bulunmadı")
```

\W
Dizenin herhangi
bir kelime
karakteri
İÇERMEDİĞİ bir
eşleşme
döndürür
(a-z, 0-9 , _
dışındaki
karakterleri çıktı
olarak verir)

```
[' ', '!', ' ', '#']
Evet,en az 1 eşleşme bulundu
```

```
import re
degisken="Yeni bir örnek"

fonksiyon=re.findall("örnek\Z",degisken)
print(fonksiyon)

if fonksiyon:
    print("Evet,minimum 1 eşleşme bulundu")
else:
    print("Hayır,hiç eşleşme bulunmadı")
```

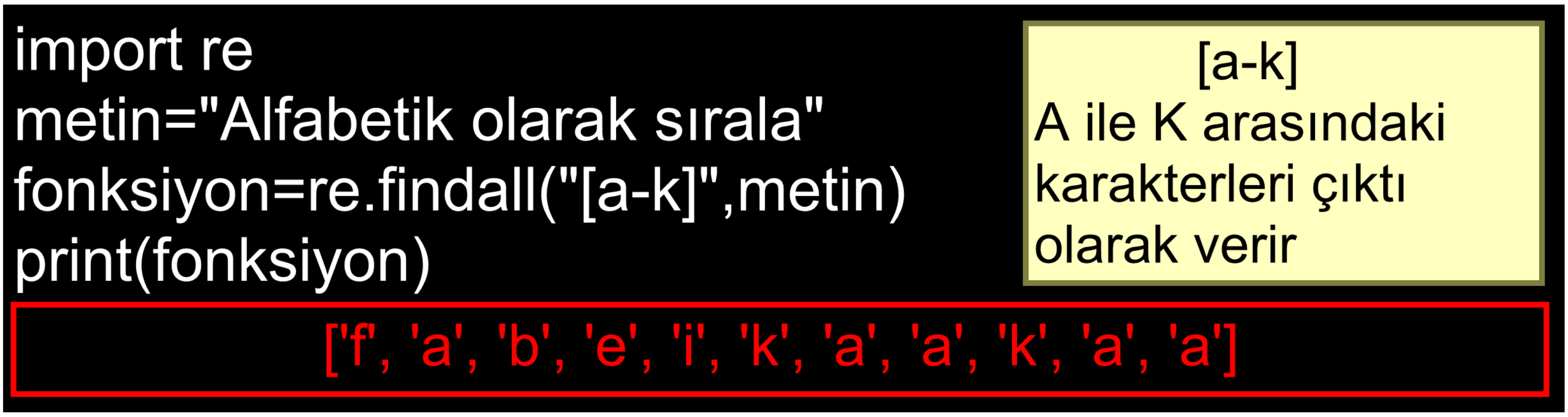
\Z
#Dizenin "örnek" ile bitip bitmediğini kontrol eder

```
['örnek']
Evet,minimum 1 eşleşme bulundu
```

```
import re

degisken="Farklı bir fonksiyon"

ara=re.findall("[fir]",degisken)

print(ara)

if ara:
    print("Eşleşme bulundu")

else:
    print("Eşleşme bulunamadı")
```

[fir]
Dizede, f i ve r karakterlerini bulur ve çıktı olarak verir

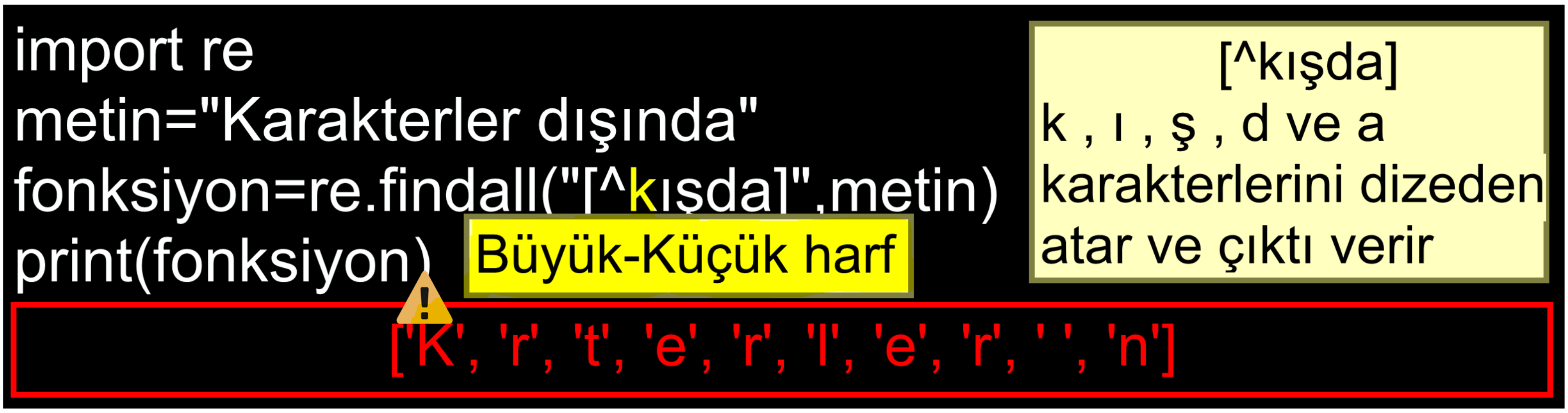
```
['r', 'i', 'r', 'f', 'i']
Eşleşme bulundu
```



```
import re
metin="Alfabetik olarak sırala"
fonksiyon=re.findall("[a-k]",metin)
print(fonksiyon)
```

```
['f', 'a', 'b', 'e', 'i', 'k', 'a', 'a', 'k', 'a', 'a']
```

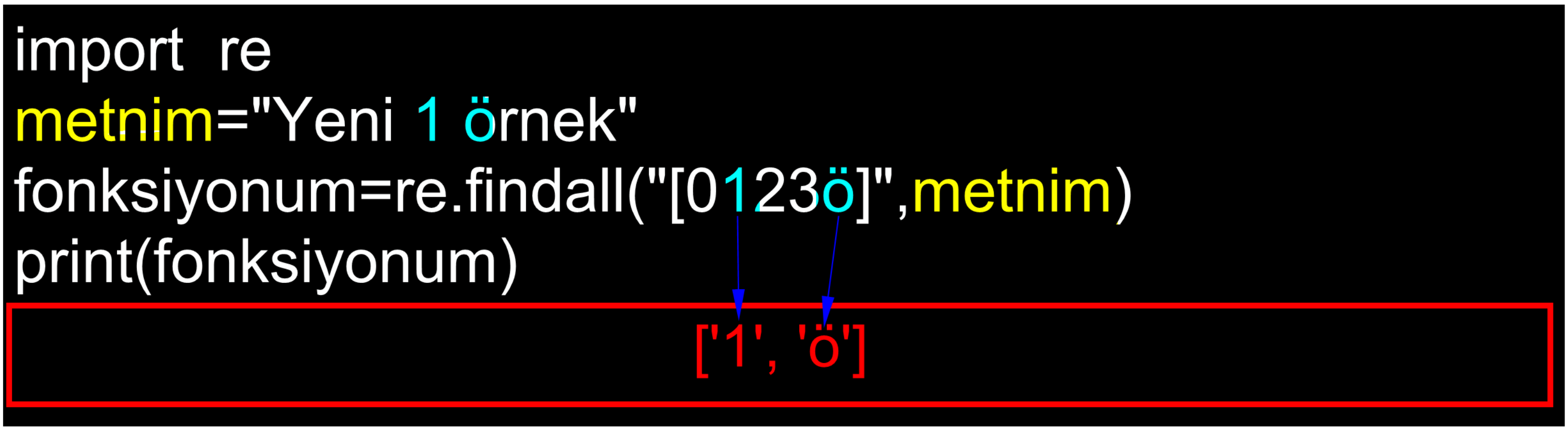
[a-k]
A ile K arasındaki karakterleri çıktı olarak verir



```
import re
metin="Karakterler dışında"
fonksiyon=re.findall("[^kışda]",metin)
print(fonksiyon)
```

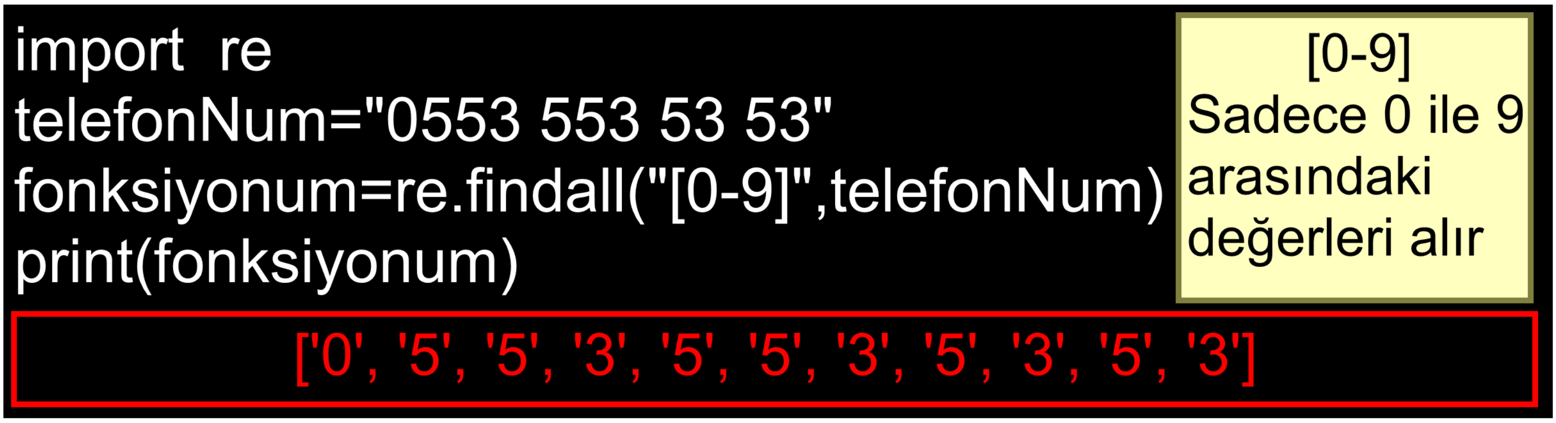
```
['K', 'r', 't', 'e', 'r', 'l', 'e', 'r', ' ', 'n']
```

[^kışda]
k , ı , ş , d ve a karakterlerini dizeden atar ve çıktı verir



```
import  re
metnim="Yeni 1 örnek"
fonksiyonum=re.findall("[0123ö]",metnim)
print(fonksiyonum)
```

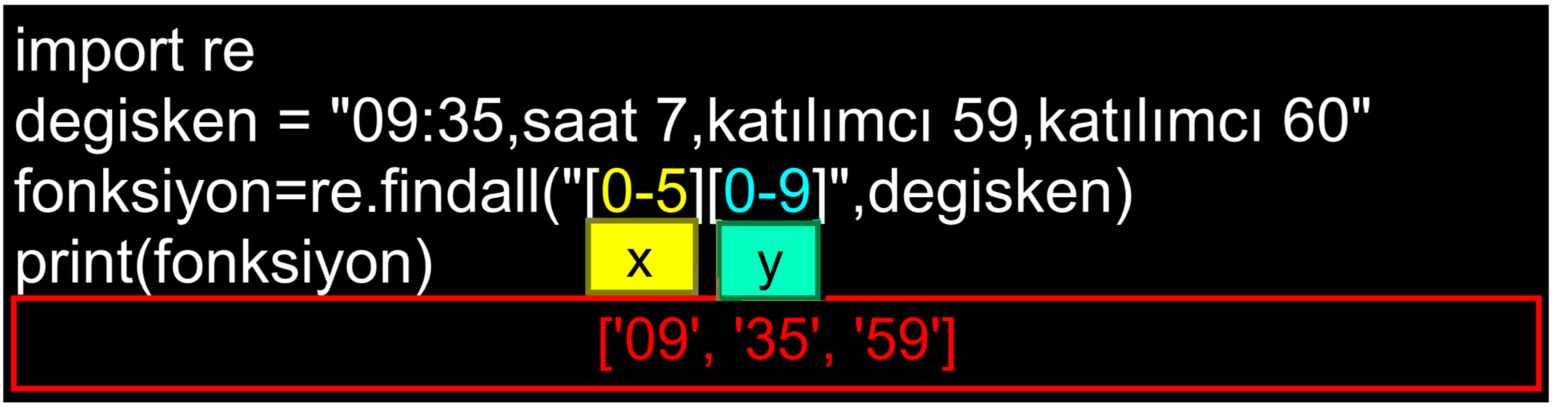
```
['1', 'ö']
```



```
import  re
telefonNum="0553 553 53 53"
fonksiyonum=re.findall("[0-9]",telefonNum)
print(fonksiyonum)
```

```
['0', '5', '5', '3', '5', '5', '3', '5', '3', '5', '3']
```

[0-9]
Sadece 0 ile 9 arasındaki değerleri alır

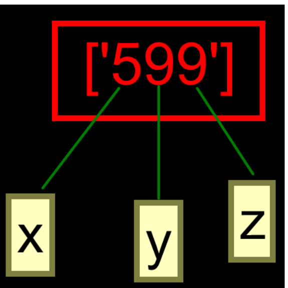


```
import re
degisken = "09:35,saat 7,katılımcı 59,katılımcı 60"
fonksiyon=re.findall("[0-5][0-9]",degisken)
print(fonksiyon)
```

```
['09', '35', '59']
```

```
import re
degisken = "09:35,saat 7,katılımcı 59,katılımcı
599,katılımcı 600"
fonksiyon=re.findall("[0-5][0-9][0-9]",degisken)
print(fonksiyon)
```

x y z



```
import re
metin1 = "BÜYÜK HARF KONTROLÜ"
metin2= "küçük harf kontrolü"

fonksiyon1=re.findall("[A-Z]",metin1)
fonksiyon2=re.findall("[a-z]",metin2)
fonksiyon=re.findall("[A-Za-z]",metin1+" "+metin2)
print(fonksiyon1," ",fonksiyon2," ",fonksiyon)
```

```
['B', 'Y', 'K', 'H', 'A', 'R', 'F', 'K', 'O', 'N', 'T', 'R', 'O', 'L']   ['k',
'k', 'h', 'a', 'r', 'f', 'k', 'o', 'n', 't', 'r', 'o', 'l']   ['B', 'Y', 'K', 'H', 'A',
'R', 'F', 'K', 'O', 'N', 'T', 'R', 'O', 'L', 'k', 'k', 'h', 'a', 'r', 'f', 'k', 'o',
'n', 't', 'r', 'o', 'l']
```

```
import  re
metin = "Aranacak karakter : +"
ara=re.findall("[+]",metin)
print(ara)
```

```
['+']
```

```
import  re

metnim="Metinde ara Metinde ara"
ara=re.search("eti",metnim)
print(ara)
```

eti

metnin içinde eti değerini arattı, bulamadığında none döndürür

```
<re.Match object; span=(1, 4), match='eti'>
```

```
import  re
degisken="Burada arayacak"
ara=re.search(r"\ba\w",degisken)
print(ara)
```

a=7.index

r : read

Belirtilen karakterlerin bir kelimenin başında veya sonunda olduğu bir eşleşme döndürür

\b

```
<re.Match object; span=(7, 9), match='ar'>
```

```
import  re
degisken="Burada arayacak"
ara=re.search(r"\ba\w",degisken)
print(ara.span())
```

```
(7, 9)
```

Önceki örnekten farklı olarak span kullanıldı

span : başlangıç ve bitiş konumlarını içeren bir sonuç döndürür

```
import re

metnimiz="Bu metinde arayacak"
fonksiyon=re.search(r"\ba\w",metnimiz)
print(fonksiyon.string)
```

string özelliği, search dizesini döndürür

```
Bu metinde arayacak
```

```
import re

metnim="Bu metinde arayacağız"

ara=re.search(r"\ba\w+",metnim)

print(ara.group())
```

```
arayacağız
```

Eşleşme bulunamassa , none değerini döndürür

```
import re

#Örnek 1 B\

metin="'\B'nin kullanımını öğreneceğiz"
fonksiyon=re.findall(r"\Beceğiz",metin)
print(fonksiyon)

#Örnek 2 B\

metin="'\B'nin kullanımını öğreneceğiz"
fonksiyon = re.findall(r"kull\B",metin)
print(fonksiyon)
```

\B
Belirtilen karakterlerin bulunduğu, ancak bir kelimenin başında (veya sonunda) OLMAYAN bir eşleşme döndürür

```
['eceğiz']
['kull']
```

```
import re

telefonNum = "0551 551 51 51"
kontrol=re.findall("\d",telefonNum)
print(kontrol)
```

\d
Dizenin rakam (0-9) içerip içermediğini kontrol eder

```
['0', '5', '5', '1', '5', '5', '1', '5', '1', '5', '1']
```

```
import re

kisi = "Şahin 0551 551 51 51"

kontrol=re.findall("\D",kisi)

print(kontrol)
```

\D
Rakam dışında her karaktere bir eşleşme döndür

```
['Ş', 'a', 'h', 'i', 'n', ' ', ' ', ' ', ' ']
```

```
import re

metin = "Her boşlukta eşleşme döndürür"
kontrol=re.findall("\s",metin)
print(kontrol)
```

\s
Her boşluk karakterinde, bir eşleşme döndürür

```
[' ', ' ', ' ']
```

```
import re

metin = "Boşlukları görmezden gelir"
kontrol=re.findall("\S",metin)
print(kontrol)
```

\S
Boşlukları kaldırarak karakterleri döndürür

```
['B', 'o', 'ş', 'l', 'u', 'k', 'l', 'a', 'r', 'ı', 'g', 'ö', 'r', 'm', 'e', 'z', 'd', 'e',
'n', 'g', 'e', 'l', 'i', 'r']
```

```
import re

metin = "0123_!#$abCD"
kontrol=re.findall("\w",metin)
print(kontrol)
```

\w
a-z
0-9
_ Alt çizgi
dışındakileri atar

```
['0', '1', '2', '3', '_', 'a', 'b', 'C', 'D']
```

```
import re

metin = "0123_!#$abCD"
kontrol=re.findall("\W",metin)
print(kontrol)
```

\W
a-z
0-9
_ Alt çizgi
dışındakileri tüm karakterleri döndürür

```
['!', '#', '$']
```

```
import re

metin = "Bu cümle neyle bitiyor"
fonksiyon=re.findall("bitiyor\Z",metin)
print(fonksiyon)
fonksiyon=re.findall("iyor\Z",metin)
print(fonksiyon)
```

```
['bitiyor']
['iyor']
```

TEBRİKLER

Temel Seviye Python kitabımızın 1.kısmını başarıyla tamamladın...

2.kısmında görüşmek üzere...

İlk sayfaya dönerek bizi
takip edebilirsiniz